VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZPRACOVÁNÍ RASTROVÉHO OBRAZU POMOCÍ FPGA

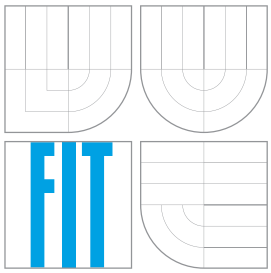
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR MUSIL

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZPRACOVÁNÍ RASTROVÉHO OBRAZU POMOCÍ FPGA

RASTER IMAGE PROCESSING USING FPGA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR MUSIL

VEDOUCÍ PRÁCE
SUPERVISOR

Doc.Dr.Ing. PAVEL ZEMČÍK

BRNO 2012

Abstrakt

Tato práce se zabývá návrhem a implementací hardwarové jednotky pro detekci objektů ve videu. Návrh jednotky je optimalizován pro rychlé proudové zpracování. Detekce objektů se provádí pomocí příznakových klasifikátorů. Jako slabé klasifikátory jsou využity lokální obrazové příznaky. Je navržena a implementována nová metoda pro detekci objektů různé velikosti. Detektor používá algoritmy pro zrychlení detekce využívající sousední pozice. Má dostatečný výkon pro souběžnou detekci až dvou různých tříd objektů. Funkčnost detektoru byla ověřena simulací a některé části byly implementovány na vývojovém kitu.

Abstract

This thesis describes the design and implementation of hardware unit to detect objects in the image. Design of unit is optimized for fast streaming processing. Object detection is performed by the trained classifiers using local image features. It describes a new technique for multiscale detection. Detector used accelerating algorithm based on neighboring positions. The correct functionality of the detector is verified by simulation and part of a whole is implemented on development kit.

Klíčová slova

Detekce objektů, detekce objektů různé velikosti, AdaBoost, obrazové příznaky, programovatelný hardware, FPGA

Keywords

Object detection, multiscale detection, AdaBoost, image features, programmable hardware, FPGA

Citace

Petr Musil: Zpracování rastrového obrazu pomocí FPGA, diplomová práce, Brno, FIT VUT v Brně, 2012

Zpracování rastrového obrazu pomocí FPGA

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc.Dr.Ing. Pavla Zemčíka

.....
Petr Musil
22. května 2012

Poděkování

Děkuji svému vedoucímu práce Doc.Dr.Ing. Pavlu Zemčíkovi za odborné vedení, konzultace a připomínky, které mi pomohly při řešení této diplomové práce. Chtěl bych také poděkovat Ing. Romanovi Juránkovi za konzultace a mnohé rady.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V dnešní době rostoucího technologického pokroku v oblasti výpočetní techniky je možné řešit i situace, které byly v minulosti známé jen z sci-fi filmů. Mezi ně patří bezesporu i možnost počítačů vidět. Moderní metody zpracování obrazu dokáží již například automaticky rozeznat osoby na fotografii a přiřadit k nim jména nebo třídit automaticky fotky v albu. Jsou schopny také detekovat různé objekty ve videu a určit například směr jejich pohybu. Pro všechny tyto úlohy počítačového vidění platí, že pro tuto činnost potřebují velké množství výpočetního výkonu. Je tedy možné je vykonávat pouze na výkonných počítačích, které jsou poměrně drahé a mají nekompaktní rozměry.

V této práci je představen návrh zařízení pro detekci objektů na programovatelném hardware. Použití programovatelného hardwaru místo klasické koncepce osobního počítače umožňuje právě tvorbu malého přenosného a velmi efektivního zařízení pro detekci objektů s velmi nízkou spotřebou, umožňující i dlouhodobý provoz z baterie. Možnosti použití takovýchto zařízení jsou široké. Nabízí se například v oblastech bezpečnostních dohledových systémů nebo v automobilovém průmyslu pro detekci chodců a dopravních značek.

Hlavním cílem diplomové práce je navrhnout, implementovat a otestovat hardwarový detektor objektů v obraze. Práce se zaměřuje na použití klasifikátorů pro detekci objektů pracujících na FPGA čipech. Jsou upraveny stávající algoritmy detekce objektů a jsou představeny nové metody pro detekci objektů různé velikosti a možnosti paralelizace hardwarových detektorů.

V následující kapitole je popsána oblast programovatelného hardware. Jsou zmíněny jeho vlastnosti a vývoj systému pro FPGA. V třetí kapitole je rozebrána detekce objektů v obraze. Jsou představeny možnosti detekce objektů. Kapitola se zaměřuje především na využití trénovaných klasifikátorů. Jsou popsány vlastnosti těchto klasifikátorů, způsob trénování a jejich vyhodnocení. Kapitola zmiňuje také možnosti akcelerace detekce objektů a detekci na hardwarových jednotkách. Ve čtvrté kapitole jsou popsány cíle práce a jsou specifikovány požadavky na navrhovaný systém. V páté kapitole je proveden návrh systému pro detekci objektů. Jsou diskutovány možné architektury, navrženy nové jednotky a rozebrány odlišnosti od detekce na procesorových systémech. Samotná realizace je popsána v šesté kapitole. Je v ní také představeno testování detektoru a jsou diskutovány jeho výsledky.

Kapitola 2

Programovatelná hradlová pole

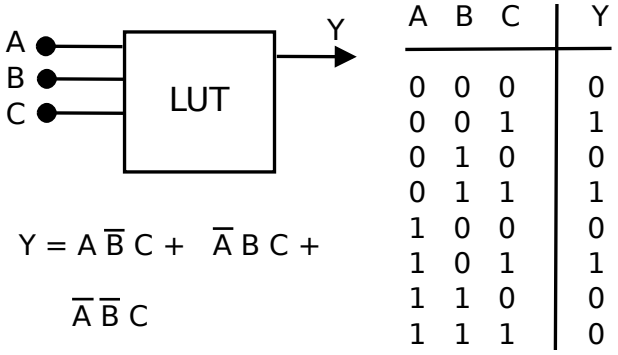
V této kapitole je stručně představena oblast programovatelného hardware se zaměřením na programovatelná hradlová pole (Field Programmable Gate Array - FPGA). Jsou popsány vlastnosti programovatelných hradlových polí, drobně zmíněn jejich historický vývoj, představena vnitřní struktura a jsou zmíněny i jazyky pro jejich popis. V kapitole je prezentován konkrétní vývojový kit, na kterém bude probíhat testování diplomové práce. V závěru kapitoly je krátce zmíněno použití programovatelného hardware pro zpracování obrazu.

2.1 Programovatelné logické obvody

Programovatelné logické obvody (Programmable Logic Device - PLD) jsou číslicové obvody, jejichž funkce není v době výroby definována. Tím se liší od standardních číslicových obvodů, jako jsou například hradla, registry nebo zákaznické obvody (Application Specific Integrated Circuit - ASIC). Struktura PLD je dána z výroby, avšak naprogramováním lze změnit propojení a parametry prvků jimiž cílová technologie disponuje, a tím ovlivnit chování celého obvodu. Zjednodušeně lze říci, že funkcionalita obvodu je dána naprogramováním.

Technologie PLD se v současné době velmi rychle vyvíjí. Dnešní PLD obvody často v rozsahu použití a výkonu předčí klasické a signálové procesory. Dosahují výkonu stovek miliónů ekvivalentních hradel, což je jednotka, udávající kolik dvouvstupových hradel NAND je možno tímto obvodem nahradit. Mezi výrobce těchto obvodů patří největší světoví výrobci polovodičových součástek jako je Xilinx, Altera, Lattice Semiconductor, Actel a Atmel. Oblast použití je také velice široká - od telekomunikací, zpracování signálů, rychlého prototypování, po kusovou výrobu složitých číslicových obvodů, u nichž by se nevyplatilo ASIC.

Jako historicky první PLD obvody je možno označit programovatelné paměti typu PROM (1970), EPROM (1971) a EEPROM(1978). U těchto obvodů lze využít paměť jako vyhledávací tabulku (Lookup Table - LUT) pro realizaci logických funkcí. LUT se používá jako generátor logických funkcí tak, že na adresové vodiče paměti je přivedena vstupní hodnota a na paměťovém výstupu je hodnota realizované funkce. Přitom je obsah paměti naprogramován tak, že na adresy jejichž hodnota odpovídá vektoru vstupních hodnot, jsou uloženy hodnoty příslušných výstupních vektorů. Výhodou použití LUT tabulky je vždy stejné zpoždění při jakékoliv vstupní kombinaci. Velkou nevýhodu byla naproti tomu neúspornost.



A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

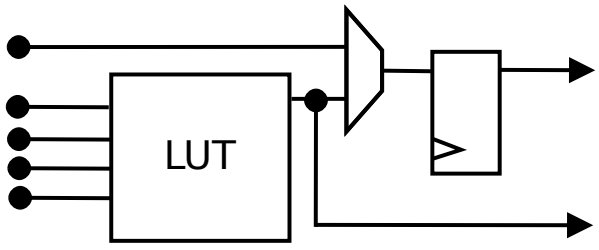
Obrázek 2.1: Použití LUT tabulky jako generátoru logické funkce

Dalším krokem byly obvody FPLA, GAL a PAL. Byly to programovatelné obvody, které nevyužívaly pro realizaci logických funkcí LUT, ale strukturu programovatelného AND pole, za kterým následovalo OR pole. Logická funkce byla tedy ve tvaru disjunktní formy. PAL obvody navíc využívají koncept výstupních makrobloků (Output Logic Macro Cell - OLMC). Tyto OLMC bylo možno naprogramovat jako standardní kombinační výstup nebo jako registr. Reprogramovatelné GAL obvody přejímají koncept OLMC. Často umožňují programování přímo na cílovém zařízení (In-System Programmable - ISP) prostřednictvím IEEE standardu JTAG. Dnešní GAL obvody mohou pracovat na vysokých frekvencích v řádu stovek MHz a jsou to velmi rozšířené polovodičové součástky.

Moderní zástupci programovatelných logických obvodů jsou CPLD (Complex Programmable Logic Device) a FPGA. CPLD obvody vycházejí z GAL obvodů, avšak obsahují navíc vnitřní propojovací pole, které umožňuje vzájemně propojit více bloků a tím vytvořit složitější a komplexnější obvod. Oproti FPGA obvodům mají výhodu v tom, že si konfiguraci ukládají do EEPROM nebo FLASH pamětí, a proto se nemusí po restartu znovu naprogramovat. Více v [19].

2.2 Programovatelná hradlová pole

FPGA obvody na rozdíl od CPLD používají pro generování logických funkcí LUT tabulky. Svoji konfiguraci si ukládají nejčastěji pomocí pamětí SRAM, a proto je nutné je po každém restartu znovu naprogramovat.



Obrázek 2.2: Zjednodušené schéma logického prvku FPGA (slice)

Základním prvkem FPGA obvodů je matice vzájemně propojených konfigurovatelných

logických bloků (Configurable Logic Block - CLB), vstupních a výstupních obvodů a specializovaných vestavěných bloků. CLB obsahují několik logických prvků (slice - podle terminologie z [26]). Tyto logické prvky se skládají z LUT tabulky, klopného obvodu a multiplexoru. Dále mohou obsahovat malé RAM paměti, posuvné registry nebo pomocnou logiku pro aritmetiku. Sousední CLB a logické prvky, které obsahují, jsou vhodně uspořádány a propojeny rychlými spoji tak, aby umožňovaly konstrukci rychlých aritmetických sčítaček a násobiček. FPGA obvody využívaly historicky čtyřvstupé LUT jednotky s jedním výstupem. Moderní obvody již často využívají šestivstupé LUT jednotky s dvěma výstupy.

Dnešní FPGA obvody obsahují velké množství speciální vestavěných obvodů. Jedná se například o paměťové bloky, rychlé hardwarové násobičky nebo přímo MAC (Multipli And Accumulate) bloky pro konstrukci filtrů, jednotky pro generování a úpravu hodinových signálů (fázové závěsy a děličky) a dokonce i přímo celé procesory. Rovněž dle zvolené konfigurace mohou obsahovat obvody pro obsluhu periferních jednotek, jako jsou například PCI Expres (PCIe Endpoint), paměťové moduly (DDR, FLASH), rychlé seriové linky (GTX Transceivery), síťovou komunikaci (IP stack) a obrazový výstup (DVI).

Paměťové bloky (Block RAM - BRAM) jsou většinou rozloženy do některých sloupců na čipu FPGA a tvoří distribuovanou paměť. Tyto bloky mohou být nakonfigurovány jako paměť ROM, RAM nebo FIFO. Paměti pak umožňují použití jako jedno nebo dvouportové s různou šířkou datového slova.

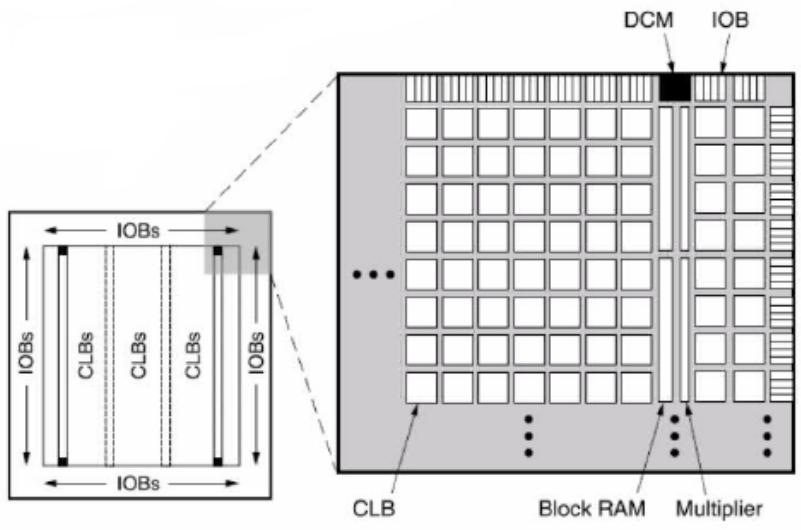
Velkou výhodou FPGA čipů je možnost dynamické rekonfigurace. Ta umožňuje změnit konfiguraci celého nebo jen části čipu a za běhu tím ovlivnit jeho výsledné chování. Této vlastnosti se využívá například u samoopravitelných zařízení nebo akceleračních jednotek, které mění svoje parametry a chování podle vstupních dat. Vhodně použitá dynamická rekonfigurace výrazně snižuje množství potřebných zdrojů, spotřebu a cenu zařízení.

Pro výrobu FPGA čipů se používají nejlepší dostupné výrobní procesy. Důvodem je pravidelná struktura čipů, na které může výrobce snadno otestovat své nové výrobní technologie. Proto bývají FPGA čipy většinou mezi prvními obvody, které sjedou z výrobní linky. Aktuálně nejnovější čipy jsou postaveny na 28 nm výrobní technologii. Obsahují skoro 2 miliony CLB a přibližně 7 miliard tranzistorů. Mohou obsahovat přes 60 Mb distribuované paměti BRAM. Více v [27]

Syntéza hradlových polí

Číslicové obvody se historicky popisují grafickou reprezentací pomocí logického schématu. Avšak s velmi rychle rostoucí složitostí číslicových systémů bylo zapotřebí vytvořit jazyky pro jejich jednodušší návrh. Proto vznikly speciální jazyky pro popis hardware (Hardware Description Language - HDL). Nejpoužívanějšími jazyky z této skupiny jsou VHDL, Verilog a System Verilog. Jejich použití zpřehledňuje návrh, umožňuje přenositelnost na jiná zařízení a hlavně výrazně zvyšuje produktivitu návrhu. Jazyky HDL rovněž umožňují simulaci, verifikaci a generování výrobních podkladů. V poslední době se objevují nové jazyky založené především na standardních vyšších programovacích jazycích. Důvodem je extrémní růst počtu dostupných hradel a snaha o zvýšení produktivity návrhu. Patří mezi ně například jazyky založené na programovacím jazyku C/C++ - SystemC, HandelC nebo CatapultC. Použití vhodných nástrojů umožňuje překlad a syntézu obvodů ze zdrojových kódů napsaných v mírně omezené podmožině jazyka C.

Syntéza logického obvodu je transformace jeho popisu a funkcionality ze zdrojových kódů nebo logických schémat do konfiguračního souboru FPGA nebo masky ASIC. Syntéza se skládá obvykle z těchto částí:



Obrázek 2.3: Obrázek struktury FPGA čipu

- Překlad podkladů (zdrojových kódů, uživatelských omezení a schémat) do popisu meziregistrových přenosů (RTL)
- Booleovské optimalizace
- Mapování do hradel nebo masky cílové technologie

Jednotlivé části překladu jsou velice složité. Často se jedná o NP-úplné problémy. Aby je bylo možné provést v rozumném čase, je nutné použít řešení pomocí heuristických funkcí, jejiž výsledek nebude nejlepší možný, ale bude se mu blížit.

2.3 Vývojový kit SP605

Pro vypracování projektu je k dispozici vývojový kit SP605 [25]. Výrobcem je společnost Xilinx, která je největší výrobní firmou v oblasti programovatelného hardwaru. Vývojový kit obsahuje FPGA čip XC6SLX45T-FGG484-3C. Jedná se o čip střední velikosti, typu Spartan 6 [26]. Architektura Spartan 6 je zaměřena do oblastí zařízení citlivých na spotřebu, s dobrým poměrem ceny a výkonu.

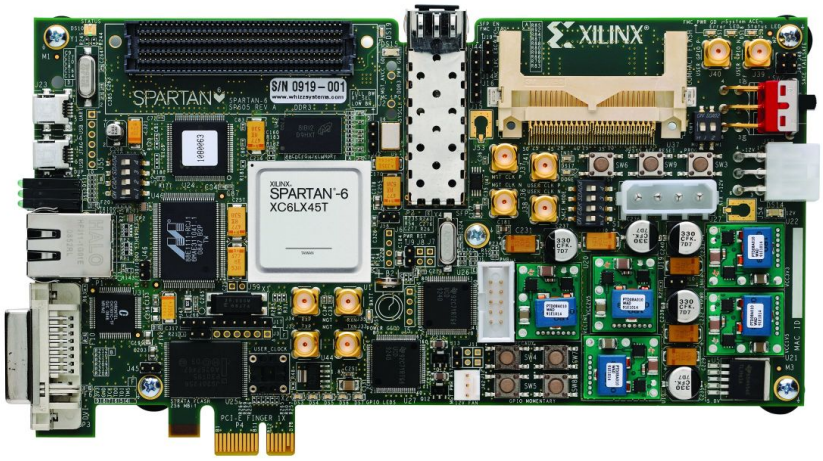
Mezi základní vlastnosti architektury Spartan 6 patří:

- Šestivstupé LUT s dvěma výstupy
- Distribuovanáné dvouportové paměti BRAM, každá o kapacitě 18Kb, s konfigurovatelnou šířkou slova
- Rychlé DSP jednotky s 18x18b násobením a 48 bitovým akumulátorem
- 45 nm výrobní proces
- Pokročilé prostředky pro řízení spotřeby, jako je hibernace a odepínání hodinového zdroje blokům čipu

- Velká podpora externích periferií jako je DDR, PCIe, LAN a jiné

Vývojový kit SP 605 je zobrazen na obrázku 2.4. Mimo samotný FPGA čip Spartan 6 obsahuje velké množství periferních obvodů:

- Dynamickou paměť DDR3 o velikosti 128 MB
- Rozhraní PCIe 1x pro komunikaci s počítačem
- DVI rozhraní pro zobrazování na monitoru
- Ethernet rozhraní s podporou 10/100/1000Mb
- JTAG rozhraní pro programování a ladění kitu
- Vysokorychlostní GTP a SFP linky



Obrázek 2.4: Vývojový kit Xilinx SP605

2.4 Zpracování obrazu na hradlových polích

Zpracování obrazu je forma zpracování signálu. Obraz je totiž prezentovaný jako 2D signál. Výstupem zpracování obrazu může být znovu obraz nebo jeho parametry a charakteristiky.

Mezi typické úlohy zpracování obrazu patří například:

- Detekce hran a geometrických primitiv
- Segmentace obrazu

- Restaurování poškozeného obrazu
- Transformace barev
- Skládání dvou a více obrázků

Zpracování obrazu je tradiční úloha, která vyžaduje velký výpočetní výkon. Je pro ni typická datová nezávislost jednotlivých bodů obrazu, umožňující poměrně jednoduchou paralelizaci výpočtu. Dalším znakem je používání datových typů s malou šířkou, typicky osm nebo deset bitů. Úlohy s těmito vlastnostmi jsou ideální pro implementaci na programovatelném hardwaru.

Použití programovatelného hardwaru umožňuje tvorbu kompexních jednotek pro zpracování obrazu. Další možností je také využití principu *Hardware/Software Codesign* a s ní související hardwarové jednotky pro předzpracování nebo akceleraci výpočtu doplněné o obecný procesor. Velkou výhodou je možnost rekonfigurace programovatelného hardware, což umožňuje měnit spektrum funkcí zpracování obrazu a tím výrazně snížit nároky na zdroje a cenu výsledného zařízení.

Použití FPGA jako akcelerační struktury k DSP procesoru je představeno například v článku [31]. Je představena komunikace mezi jednotkami a několik algoritmů zpracování obrazu a počítačové grafiky, jež jsou použity pro tento systém. Použití komplexní architektury pro zpracování obrazu na FPGA je ukázáno například v [18], kde je použito pro detekci hran v obraze.

Kapitola 3

Detekce objektů v obraze

Tato kapitola stručně představuje možnosti detekce objektů v obraze. Zaměřuje se detailně na použití a trénování klasifikátorů pro tuto detekci. Je představen algoritmus AdaBoost a další algoritmy pro trénování klasifikátorů, která z něj vycházejí. V kapitole jsou popsány také obrazové příznaky, které se používají jako slabé klasifikátory. V závěru kapitoly jsou zmíněny některé možnosti akcelerace detekce objektů klasifikátory a jsou detailně popsány existující varianty hardwarových detektorů.

3.1 Přístupy k detekci objektů

Úkolem detekce objektů v obraze je zjistit, zda se v daném obraze vyskytují objekty zájmu a pokud ano, určit co nejpřesněji jejich pozici, rozměry a natočení. Samotné nalezení objektu je ale netriální krok, který je navíc velice náročný na výkon. Důvodem je velké množství pozic, na kterých je potřeba hledat objekty a značná variabilita ve velikosti a natočení objektů nebo úhlu pohledu kamery.

Pro detekci objektů se využívá několik metod:

- *knowledge-based* - Jsou založeny na vědomostech o hledaném objektu. Například pro detekci obličejů se může jednat o barvu kůže, tvar a symetrie očí nebo vzájemnou pozici nosu a úst. Do této skupiny patří metody rozpoznávání shora dolů(top-down)[28] a zdola nahoru(bottom-up)[2].
- *template matching* - Jsou založeny na hledání míry podobnosti vstupního obrázku s vytvořenými modely. Tyto modely mohou popisovat například tvar, barvu nebo texturu. Nevýhodou těchto metod je jejich nízká odolnost proti variabilitě objektů. Tuto vlastnost řeší některé úpravy založené na detekci podvzorů nebo deformaci modelů představené v [29].
- *Appearance-based* - Jsou postaveny na technikách statistické analýzy a strojového učení. Na základě anotované množiny trénovacích dat vytváří klasifikátor, kterým se daný obraz postupně zpracovává. Typickými přestaviteli jsou AdaBoost (3.3), neuronové sítě nebo Support Vector Machines (SVM) [4].

V dnešní době se nejčastěji používají metody Appearance-based, které dosahují lepších výsledků než klasické "učitelem učené"metody. Pomocí Appearance-based algoritmů strojového učení se vytvářejí klasifikátory, které jsou velmi rychlé, s dobrou schopností generalizace a vysokou úspěšností detekce. Nevýhodou může naopak být potřeba velkého množství anotovaných trénovacích dat a omezení počtu hledaných tříd objektů.

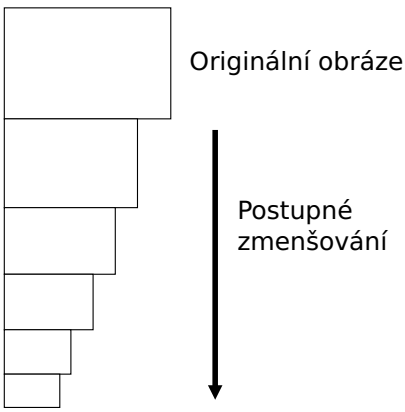
3.2 Detekce objektů v obraze klasifikátory

Klasifikátor je matematický nástroj, který slouží k třídění informací. Umožňuje třídění vstupních dat do konečného počtu tříd na základě klasifikačního algoritmu. Tento algoritmus může využívat jednoduchá třídící pravidla nebo komplexní funkce strojového učení. Nejčastěji se používá lineární klasifikátor, který provádí třídění do dvou tříd. Pro třídění do více počtů tříd je tedy nutné použít více lineárních klasifikátorů. Více v [15].

Pro vytvoření klasifikátoru pro detekci objektů je potřebná anotovaná množina trénovacích dat, která obsahuje velké množství pozitivních (hledané objekty) a negativních vzorů (pozadí). Rozměry všech vzorů by měly být stejné. Tyto rozměry odpovídají velikosti skenovacího okna. Pro detekci se většinou používá čtvercové okno. Jeho rozměry přitom odpovídají minimální velikosti objektu, který jsme schopni klasifikátorem najít. Velikost okna ovlivňuje dobu trénování. Z tohoto důvodu by měla být pokud možno co nejmenší, avšak s ohledem na zachování dostatečného rozlišení detekovaného objektu. V praxi se nejčastěji používají okna o velikostech od 16 do 31 obrazových bodů podle typu detekovaného objektu.

Samotná detekce objektů probíhá nad částmi obrazu. Detektor využívá klasifikátor, který se pohybuje po obraze a na každé jeho pozici určí, zda se na ní nachází nebo nenachází hledaný objekt. Počet těchto pozic je velký a odpovídá velikosti obrazu. Proto by měl být klasifikátor co možná nejrychlejší, aby byl schopen všechny tyto pozice vyhodnotit v rozumném čase. Problematické je také hledání objektů různých velikostí. Klasifikátor je totiž natrénovaný pouze na okno určité velikosti a je tedy schopen nacházet v obraze objekty odpovídající této velikosti. Pro nalezení větších objektů je nutné zvětšit velikost okna klasifikátoru nebo zmenšit vstupní obraz. V praxi se skoro vždy zmenšuje vstupní obraz. Pro možnost nalezení objektů o různé velikosti se vytváří takzvaná pyramida obrazů (obrázek 3.1), která obsahuje původní obraz a několik jeho podvzorkovaných verzí se snižujícím se rozlišením. V praktické realizaci se nejčastěji používá zmenšení měřítka s krokem kolem 1.2. Velikost tohoto měřítka je odpozorována a souvisí s generalizační schopností klasifikátoru viz [23].

Podobné problémy jako u detekce objektů různé velikosti nastávají i u detekce natočených objektů. Klasifikátor je opět natrénovaný jen na určité natočení objektů. Pro detekci různě natočených objektů je tedy nutné vytvořit několik verzí obrazu s různým natočením.



Obrázek 3.1: Pyramida obrazů s různým měřítkem

Avšak u mnoha aplikací neni nutné natočení řešit, protože můžeme předpokládat, že se detekované objekty v obraze budou vyskytovat pouze s jedním natočením. Příkladem může být například detekce obličeje, kde můžeme z anatomie člověka předpokládat, že se bude hlava vyskytovat ve vzpřímené poloze.

3.3 Algoritmus AdaBoost

Algoritmus AdaBoost byl představen v roce 1995 [5]. Jedná se o algoritmus strojového učení založený na metodě boostingu. Vhodně spojuje slabé klasifikátory a vytváří nový silnější klasifikátor, který je přesnější než všechny použité slabé klasifikátory. Jeho výhodou je schopnost exponenciálně snižovat chybu výsledného klasifikátoru. Tato vlastnost umožňuje trénovat v relativně krátké době klasifikátory s nízkou chybou.

Použité slabé klasifikátory mohou mit i špatnou klasifikační schopnost, alespoň nad hodnotou odhadu (chyba klasifikace tedy musí být menší než 0.5). Tato vlastnost dovoluje tvorbu jednoduchých a rychle vyhodnotitelných klasifikátorů, jejichž použití v reálných podmínkách umožňuje výslednou klasifikaci v krátkém čase. Tyto vlastnosti umožňují použití těchto klasifikátorů například pro detekci objektů ve videu v reálném čase [23].

Samotný algoritmus AdaBoost má za cíl vhodně vybrat z množiny všech možných klasifikátorů podmožinu slabých klasifikátorů $H, h_t(x) \in H$, které dokáží nejlépe rozlišovat do určených tříd. Výsledný silný klasifikátor $H(x)$ je nelineární funkce, která vzniká lineární kombinací klasifikátorů $h_t(x)$ z podmožiny H a jejich vah α_t, které jsou určeny z chyby jejich klasifikace. Vzorec 3.1 je zápis silného klasifikátoru pro $t = 1, 2..T$, kde T je množina slabých klasifikátorů.

$$H(x) = sign(\sum_{t \in T} \alpha_t h_t(x)) \tag{3.1}$$

Vstupem algoritmu AdaBoost je ohodnocená trénovací množina (x_i, y_i), kde $x_i \in X$ je prvek množiny trénovacích vzorků a $y_i \in Y$ je jeho ohodnocení pro $i = 1, 2, ..M$ (kde M je počet vzorků trénovací sady). Prvky množiny Y mohou nabývat hodnot $Y \in \{-1, 1\}$. Hodnotu 1 pro hledané objekty a -1 pro protipřípady. Algoritmus si uchovává distribuci chyby trénovacích dat $D_t(i)$, kterou na počátku inicializuje na stejnou hodnotu pro všechny prvky trénovací množiny. V každé iteraci trénování t se distribuce $D_t(i)$ upravuje. Roste prvkům trénovací množiny, které se nedaří dlouhodobě správně klasifikovat. Díky této vlastnosti se algoritmus snáze přizpůsobuje obtížně klasifikovatelným vzorům.

$$\epsilon_t = \sum_{t \in T} D_t(i) \tag{3.2}$$

$$\alpha_t = \frac{1}{2} \ln(\frac{1 - \epsilon_t}{\epsilon_t}) \tag{3.3}$$

Z sumy distribuce chyby jednotlivých prvků trénovací množiny se vypočítá chyba klasifikátoru ϵ_t podle 3.2. Algoritmus vybere klasifikátor s nejmenší chybou a z ní jednoduše určí váhu klasifikátoru α_t dle 3.3. Pokud je chyba ϵ_t menší než 0.5, což je nutná podmínka AdaBoostu, bude chyba výsledného klasifikátoru v čase $t + 1$ menší nebo rovna chybě v čase t. Chyba výsledného klasifikátoru na trénovacích vzorech tedy konverguje k nule. Tato vlastnost však může být i nevýhodou, protože může vést k tvorbě přetrénovaného klasifikátoru, který ztratil schopnost generalizace. Avšak u AdaBoostu tento problém prakticky nenastává.

Mějme $S = \langle (x_1, y_1), ..., (x_m, y_m) \rangle$, $x_i \in X, y_i \in Y = \{1, -1\}$

Inicializuj $D_1(i) = \frac{1}{m}$

pro $t = 1, ..., T$:

Natrénuj slabé klasifikátory na distribuci $D_t(i)$
(najdi takové parametry klasifikátorů, které poskytují nejmenší chybu)

Vyber klasifikátor $h_t : X \to \{-1, 1\}$ s nejmenší chybou

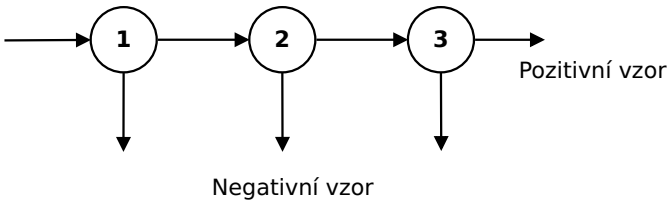
Vypočítej váhu $\alpha_t = \frac{1}{2} \ln(\frac{1 - \epsilon_t}{\epsilon_t})$

Aktualizuj $D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$, kde Z_t je normalizační faktor

Obrázek 3.2: Originální verze algoritmu AdaBoost popsaná pseudokódem převzatá z [5].

Viola-Jones

Modifikovanou verzi algoritmu AdaBoost představili v roce 2001 autoři Viola a Jones v [23] a použili ji pro natrénování klasifikátoru pro detekci obličejů. Tato verze zavádí takzvané kaskádové zapojení klasifikátoru, které snižuje průměrnou dobu prohledávání okna a tím výrazně urychluje průběh detekce.



Obrázek 3.3: Kaskádové zapojení Viola-Jones klasifikátoru

Myšlenka použití kaskádového zapojení vychází z pozorování, že i klasifikátor obsahující malé množství příznaků je schopný vybrat téměř všechny pozitivní vzorky a k nim jen malé množství negativních vzorků. Algoritmus tedy vytvoří kaskádu krátkých, rychle vyhodnotitelných klasifikátorů, které pustí do další fáze detekce téměř všechny pozitivní případy a méně než polovinu negativních. Ostatní jsou ohodnoceny jako negativní a dále nezpracovávány. Pro určení, zda se objekt v daném okně skutečně nachází, se musí vyhodnotit celá kaskáda klasifikátorů. Statisticky je ale počet negativních případů v obraze mnohem větší než pozitivních, a proto se vyplatí nezpracovávat negativní případy.

WaldBoost

Algoritmus WaldBoost představili Šochman a Matas v [21]. Oproti kaskádě klasifikátorů je mnohem obecnější. Kombinuje AdaBoost s Waldovou strategií sekvenčního testu poměru pravděpodobností (SPRT). To umožňuje vytvářet klasifikátory, které mohou po dosažení požadované přesnosti detekce předčasně zamítnou nebo příjmout pozici bez nutnosti vyhodnocení celého klasifikátoru a tím urychlit průběh celé detekce. Požadovaná přesnost je určena dvojicí chybových metrik, které slouží jako parametr pro trénování. Často je ale určena jen metrika pro zamítnutí pozice, proto musí být pro pozitivní detekci vyhodnocen klasifikátor celý.

3.4 Slabé klasifikátory

V předchozí podkapitole bylo popsáno, jak algoritmus AdaBoost vybírá slabé klasifikátory a vytváří nový silný klasifikátor. Slabým klasifikátorem může být obecně libovolná funkce nad oknem obrazu. Jedinou podmínkou je, aby chyba její detekce byla menší než 0.5, tedy byla lepší než je chyba odhadu. Původní algoritmus počítal s použitím několika složitých slabých klasifikátorů, které mají samy dobrou klasifikační schopnost. Tyto složité klasifikátory jsou ale velice složité, a proto je jejich vyhodnocení většinou časově příliš náročné. Proto se pro detekci objektů v obrazu používají nejčastěji jednoduché, velmi rychlé klasifikátory.

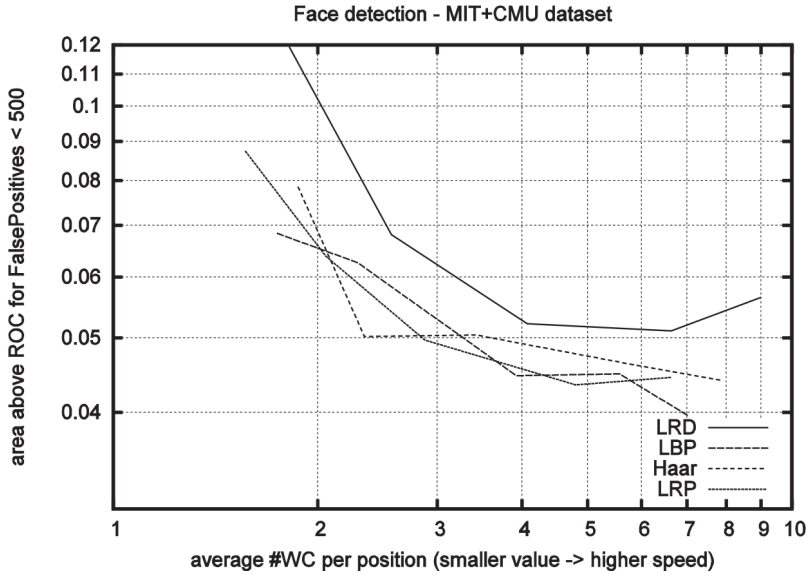
Určitě nejjednoduší by bylo použít jako klasifikátor přímo hodnoty pixelů. Ale počet takovýchto klasifikátorů na skenovacím okně je poměrně malý. Statisticky se však ve větším počtu klasifikátorů bude vyskytovat více klasifikátorů s menší chybou. Algoritmus Adaboost vybírá právě klasifikátory s nejmenší chybou, kterých bude ve větším počtu klasifikátorů více, proto i výsledný silný klasifikátor bude mít menší chybu detekce. Navíc by takové klasifikátory nebyly invariantní vůči změnám osvětlení. V praxi se tedy většinou používají takzvané *obrazové příznaky* (features)[8], která jsou flexibilnější a generují větší počet slabých klasifikátorů na skenovacím okně. Nejpoužívanější obrazové příznaky pro detekci objektů v obraze jsou:

- *Haarovy příznaky*
- *Local Binary Patterns*
- *Local Rank Functions*
- *Histogram orientovaných gradientů*

Pro detekci můžeme použít i jiné příznaky, jako jsou například Gáborovy vlnky nebo jiné vlnkové a lineární transformace. Jedná se však o příliš komplexní funkce, jejiž vyhodnocení by bylo časově velice náročné.

Haarovy příznaky

Haarovy příznaky jako klasifikátor pro detekci objektů představili Viola a Jones v [23]. Vycházejí z Haarových vlnek, které matematicky popsal v roce 1909 Alfred Haar [6]. Na obrázku 3.5 jsou zobrazeny některé typy používaných příznaků. Hodnota samotného příznaku je dána rozdílem součtu pixelů nad bílou částí a součtu pixelů nad černou částí, jak je uvedeno v 3.4.



Obrázek 3.4: Porovnání obrazových příznaků použitých pro detektor obličejů. Převzato z[8].

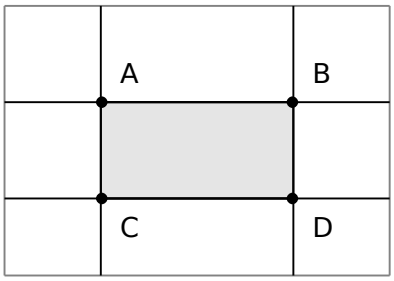
$$f(x) = \sum_{w \in W} x(w) - \sum_{b \in B} x(b) \tag{3.4}$$

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{3.5}$$

Výpočet Haarových příznaků je založen na sumách pixelů v obdelníkové oblasti. Přímý výpočet pomocí součtu hodnot jednotlivých obrazových bodů není příliš efektivní, protože doba jeho výpočtu roste s velikostí oblasti příznaku. Mnohem vhodnější je pro výpočet Haarova příznaku použít takzvaný integrální obraz. Integrální obraz je reprezentace původního obrazu, která neukládá přímo hodnoty obrazových bodů, ale jejich součet v oblasti vlevo a nahoru od původního obrazového bodu jak popisuje vzorec 3.5. Výpočet Haarova příznaku nad integrálním obrazem pak probíhá v konstantním čase. Pokud jsou A, B, C, D hodnoty integrálního obrazu rohových bodů oblasti jak naznačuje obrázek 3.6, pak je součet obrazových bodů původního obrazu v oblasti roven $A + D - B - C$.

Haarovy příznaky se velmi často používají jako klasifikátory pro detekci objektů, například v [23]. Jejich výhodou je invariance vůči změnám osvětlení a také velký počet možných příznaků nad oknem klasifikátoru. Diky použití integrálního obrazu je jejich výpočet velmi rychlý a navíc v konstantním čase. Mezi nevýhody Haarových příznaků patří nutnost normalizace integrálního obrazu při změně měřítka viz [8] a také nutnost použití datového typu s větším rozsahem pro uchování hodnot integrálního obrazu. Použití normalizace platí obecně pro všechny příznaky založené na vlnkách. Je možné ji velice efektivně řešit na CPU, avšak je problematická při použití GPGPU nebo hardwarových detektorů, kde je nutné přepočítat integrální obraz zmenšených měřítek znovu. Rovněž nutnost použití datových typů s větším rozsahem limituje použití Haarových příznaků na hardwarových detektorech.

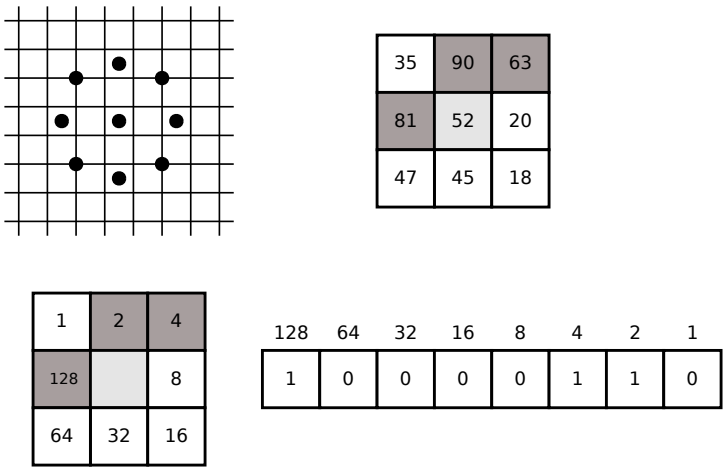
Obrázek 3.5: Tvary některých používaných Haarových příznaků.



Obrázek 3.6: Použití integrálního obrazu pro výpočet Haarových příznaků.

Local Binary Patterns

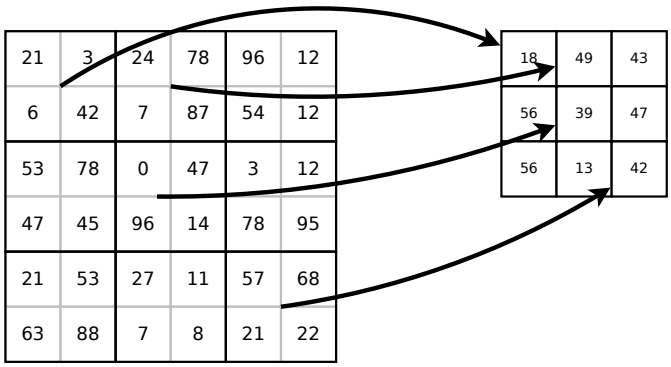
Local Binary Patterns (LBP) byly představeny v roce 2000 [17]. Jsou používány ke zpracování textury obrazu. Lze je využít například v oblasti průmyslu pro rozpoznávání defektů vzniklých při svařování. LBP zachycují lokální strukturu textury pomocí porovnání hodnoty středového bodu s jeho okolím, jak je znázorněno na obrázku 3.7.



Obrázek 3.7: Příklad výpočtu LBP příznaků. Nahoře vlevo je ukázka vzorkování okolních obrazových bodů, pravý obrázek přibližuje porovnání se středovým bodem, spodní levý použitou masku a na pravém je výsledná hodnota příznaku.

Typicky používají LBP příznaky pro výpočet osm okolních bodů. Ale existují i varianty se okolím o velikosti 4, 12, 16 i více prvků. Další variantou mohou být takzvané uniformí LBP popsané v [17]. Tyto příznaky jsou invariantí vůči rotaci. Další možností jsou také

LBP příznaky které nevyužívají přímo hodnoty jednotlivých bodů, ale hodnotu odezvy obrazu s konvolučním jádrem. Tím roste počet příznaků nad skenovacím oknem a teoreticky klesá chyba detekce. Zároveň se ale nezvyšuje množství potřebné paměti pro uložení vah klasifikátorů. V [20] jsou použity například konvoluční jádra 1×1, 1×2, 2×1 a 2×2.



Obrázek 3.8: Použití konvolučního jádra 2×2 používaného pro výpočet obrazových příznaků.

Obecně jsou LBP příznaky vhodné pro použití při detekci objektů v obraze, což je ukázano například v [8] a [24] . Jsou invariantní vůči změnám osvětlení a jejich výpočet je poměrně rychlý a navíc v konstantním čase. Struktura výpočtu umožňuje jejich efektivní použití na hardwarových detektorech.

Local Rank Functions

Příznaky založené na Local Rank Functions (LRF) jsou popsány v [10] a [32]. Jedná se o takzvané *Local Rank Differences* (LRD) a *Local Rank Patterns* (LRP). Tyto příznaky jsou založeny na myšlence, že informace v obraze může být dobře reprezentována pořadím hodnot (intenzity) obrazových bodů. Podobně jak u LBP nemusí být použíta přímo hodnota bodů v obrazu, ale i jejich odezva s konvolučním jádrem. Výpočet předpokládá určení pořadí prvků v množině M, která obsahuje zkoumaný obrazový bod a jeho lokální okolí. Formálně je postup pro určení pořadí zapsán v 3.6. Výpočty jednotlivých příznaků pro dvojice bodů jsou vyjádřeny v (3.7 a 3.8). Na obrázku 3.9 je ukázán výpočet příznaku na konkrétních hodnotách.

$$R_k = \sum_{i=1}^{n} \begin{cases} 1 & pokud\, M_k < M_i \\ 0 & jinak \end{cases} \tag{3.6}$$

$$LRD(a,b) = R_a - R_b \tag{3.7}$$

$$LRP(a,b) = R_a n + R_b;\ a,b \in \{1,...,n\} \tag{3.8}$$

Příznaky založené na LRF jsou obecně velmi dobře použitelné pro detekci objektů v obraze, jak je ukázáno například v [8] a [9]. Jsou invariantní vůči změnám osvětlení, striktně lokální, umožňují rychlý výpočet v konstantní čase a jsou velice vhodné pro nasazení v hardwarových detektorech.

35	90	63
81	52	20
47	45	18

2	8	6
7	5	1
4	3	0

$$LRD(1,5) = 2 - 5 = -3$$

$$LRP(1,5) = 2 * 9 + 5 = 23$$

Obrázek 3.9: Ukázka výpočtu LRF příznaků. Nahoře vlevo je zobrazeno okolí bodu, vpravo je vypočteno pořadí prvků z tohoto okolí. Dole jsou pro ukázku vypočteny konkrétní příznaky pro dvojici levý horní a středový bod.

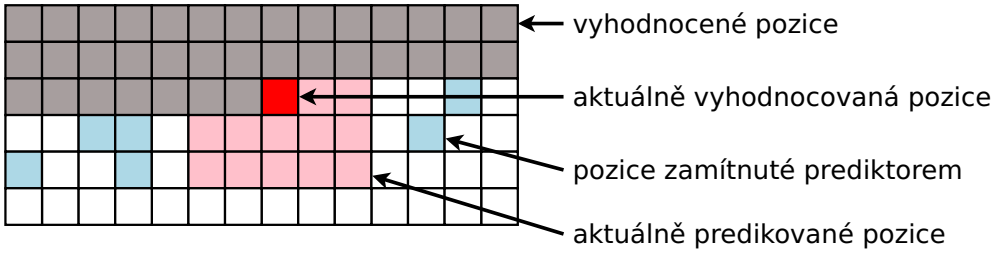
3.5 Akcelerace detekce objektů

Detekce objektů v obraze si klade veliké požadavky na výkon výpočetního systému. Použití kaskády klasifikátorů nebo klasifikátoru trénovaného pomocí algoritmu WaldBoost tyto požadavky výrazně snižuje, ale i přesto mohou být pro některé aplikace příliš vysoké. Proto se objevilo několik postupů, které si kladou za cíl tuto detekci dále urychlit. Jedná se například o metody, které využívají informace ze sousedních detekovaných pozic nebo o úpravy algoritmu pro nasazení na paralelní výpočetní struktury modernich procesorů, grafických karet nebo specializovaného hardware.

Využití sousedních pozic

Možnost využití informace z okolních pozic pro zrychlení detekce objektů v obraze představil Zemčík a kolektiv v [33]. Toto řešení využívá toho, že informace extrahovaná z jedné pozice obrazu může být použita i pro rozhodnutí okolních pozicích. Klade si za cíl zrychlit detekci negativně detekovaných vzorů. Samotný algoritmus je postaven na predikční funkci, která předpovídá odezvu klasifikátoru na blízkých okolních pozicích. To dovoluje přeskočit vyhodnocování tohoto klasifikátoru na takových pozicích, kde je predikce dostatečně jistá že neobsahují hledaný objekt. Samotnou predikční funkci můžeme nahradit predikčním klasifikátorem, který využívá stejných obrazových příznaků, jež jsou počítány původním detekčním klasifikátorem. Toto je výpočetně velmi efektivní, avšak ukládá nutnost uložení tabulek s váhou slabého klasifikátoru α_t a prahu pro každou predikovanou pozici, což ale není na většině výpočetních systému problematické.

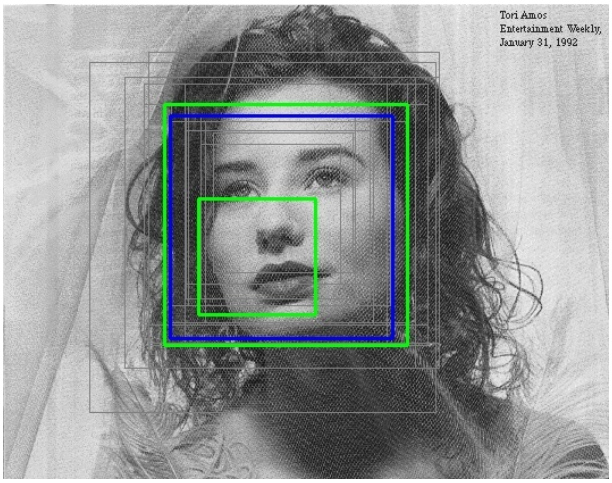
Při samotné realizaci využili autoři predikci 12 okolních bodů, jak je naznačeno v obrázku 3.10. Pro natrénování detekčního a predikčního klasifikátoru použili algoritmus WaldBoost. Výsledný detektor byl v závislosti na testovacím obrazu až čtyřikrát rychlejší než detektor, který nevyužívá predikce ze sousedních pozic a to jen při nepatrném zvýšení chybovosti detekce. V některých případech tak bylo možné dosáhnout vyhodnocení v průměru i méně než jednoho obrazového příznaku na jednu obrazovou pozici.



Obrázek 3.10: Rozložení sousednich pozic pro urychleni detekce objektů použité v [33].

Potlačení nemaximálních pozic

Další možnost zrychlení detekce představil Herout a kol. v [7]. Využívá toho, že většina aplikaci pro lokalizaci objektů vyžaduje výběr pozice v obraze s nejvyšší odezvou detekčního klasifikátoru. Při standartní detekci objektů označí klasifikátor jako detekované objeky nejčastěji několik blízkých pozic v obraze, a to jak ve vzdálenosti, tak i měřítku jak je ukázáno například na obrázku 3.11. Výsledná pozice lokalizovaného objektu je poté určena podle klasifikátoru s nejsilnější odezvou a ostatní pozice jsou zamítnuty. Metoda potlačení nemaximálních pozic si klade za cíl včasně potlačit tyto následně zamítnuté pozice, a to na základě neúplné informace částečně vyhodnoceného klasifikátoru. Tím je snížena výpočetní náročnost pro detekci kladně klasifikovaných vzorů a současně je vybrána přímo pozice s nejsilnější kladnou odezvou klasifikátoru.



Obrázek 3.11: Ukázka kladně detekovaných pozic klasifikátoru pro detekci obličejů. Modře jse označena výsledná pozice s nejsilnější odezvou. Zeleně jsou pro příklad zvýrazněny některé pozitivní detekce v blízkém měřítku. Šedou barvou jsou označeny všechny zbývající pozitivní pozice detekce.

Autoři představili několik strategií pro potlačení nemaximálních pozic. Je také představena optimální strategie, které využívá nově popsané varianty Waldovy sekvenční strategie. Navržený detektor, využívající potlačení nemaximálních pozic, je podobný kaskádě klasifi-

kátorů používaných pro detekci objektů. Výsledky ukazují, že při detekci objektů je výrazně redukováno množství výpočtů v případě lokalizace pozice, při zachování nízké chyby dané hranicí při trénování.

Využití paraelních výpočetních struktur

Standardní detekce objektů pomocí klasifikátorů je datově nezávislá v rámci jednotlivých pozic obrazu. Tato vlastnost umožňuje poměrně snadné použití paralelních výpočetních struktur pro vyhodnocení jednotlivých pozic a tím i možnost zvýšení rychlosti detekce objektů v rámci celého obrazu. Objevilo se několik konkrétních řešení využívajících paralelních jednotek moderních výpočetních systémů.

Využití replikovaných funkčních jednotek moderních procesorů v podobě SIMD představili například Herout a kol. v [8]. Vytvořili velmi efektivní detektor objektů využívající LRP obrazové příznaky, pracující na procesoru s SIMD rozšířením pomocí instrukční sady SSE. Jako výkonově kritické části algoritmu pro detekci objektu byly určeny přístup do paměti a vyhodnocení samotného příznaku. K urychlení přístupu do paměti byly použity předvypočtené pruhy obrazu k uložení odezvy obrazu s konvolunčními jádry používanými pro výpočet obrazových příznaků. Rovněž byl minimalizován přístupu do hlavní paměti a zarovnáno načítaní ve vyrovnávací paměti. Funkce pro výpočet příznaku byla upravena a přepsána do asembleru s využitím instrukcí SSE. Těmito úpravami bylo dosaženo přibližně pětinásobného zrychlení detekce oproti standardní implementaci.

Další možnost akcelerace detekce objektů pomocí paralelních výpočetních struktur byla představena v [8] a [20]. Tyto řešení využívají GPGPU výpočtů na moderních grafických kartách. Každé výpočetní jádro grafické karty zpracovává jednu pozici v obraze a tím je možno teoreticky využít plného výpočetního potenciálu. Problém však nastává u divergence jednotlivých vláken v rámcí warpu, protože počet klasifikátorů, který se musí vyhodnotit na sousedních pozicích není většinou stejný. Tento problém je řešen čekáním již vyhodnocených pozic nebo přeskládáním v rámci bloku. Autoři představili dvě implementace využívající LRP příznaky napsané pomocí CUDA a GLSL. Obě implementace dosahují srovnatelných výsledků, na testovaném hardware(Intel Core2 Duo E8200 a NVidia GTX280) jsou zhruba 25 krát rychlejší než při použití standardní implementace na CPU.

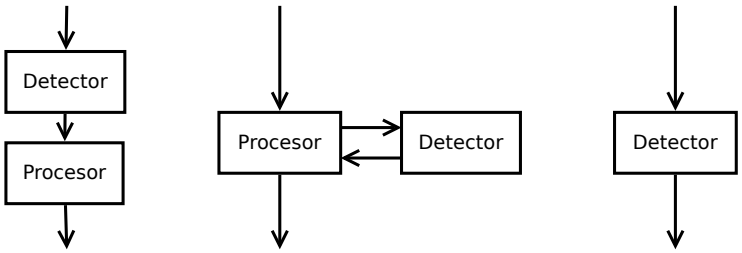
3.6 Hardwarová detekce objektů

Detekce objektů muže být velmi efektivně vykonávána na hardwarových jednotkách, jak je ukázáno například v [12], [13], [14], [22] nebo [34]. Principy algoritmů používaných v hardwarových detektorech jsou obecně podobné jako u detektorů pracujících na obecném procesoru. Rozdíly vycházejí především z výhodných vlastností hardware, jako je možnost masivního paralelizmu, snadná manipulace s jednotlivými bity nebo variabilní šířka datových slov. Na druhou stranu jsou zohledněny i nevýhody, jako je nižší výpočetní frekvence, výsoká náročnost některých matematických operací (děleno, odmocnina) nebo omezení paměťové struktury.

Obecně je možné použít hardwarové detektory jako:

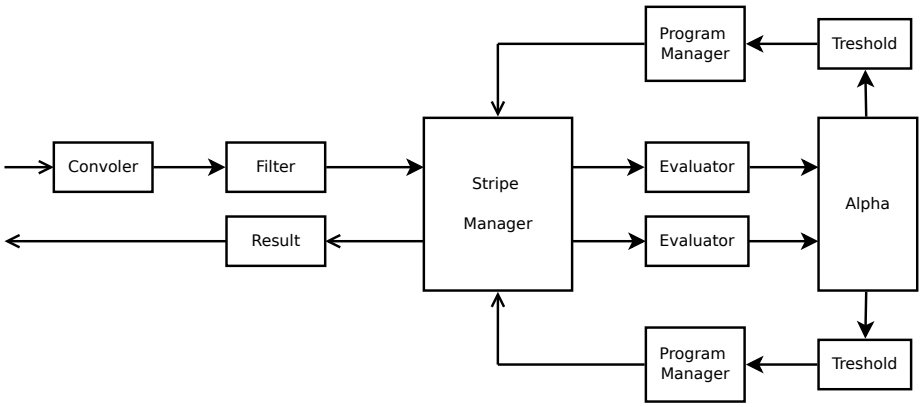
- *rychlé předzpracovací jednotky* - Jsou používány především ve výkoných systémech. Jejich úkolem je rychle vybrat několik potenciálních pozic objektů zájmu, které jsou poté dále zpracovány a přesně vyhodnoceny jiným systémem.

- *jednotky s hardwarovým detektorem jako koprocesorem* - Jsou používány hlavně v úsporných zařízeních. Většinou se jedné o rychlé jednotky pro vyhodnocení obrazových příznaků, které slouží pro akceleraci detekce pomocí obecného procesoru.
- *komplexní jednotky pro detekci objektů* - Jedná se o univerzální jednotky, jež jsou schopny samostatně detekovat objekty v obraze.



Obrázek 3.12: Použití hardwarových detektorů. V pravo jako předzpracovací jednotka, uprostřek jako koprocesor a v levo jako komplexní detektor.

Mnohé hardwarové jednotky pro detekci objektů [1], [3], [11] a [30] využívají jako slabé klasifikátory Haarovy příznaky. To však neni optimální, a to z důvodu potřeby velké šířky datového slova pro uložení integrálního obrazu a také z nutnosti načíst čtyři hodnoty z náhodných míst v paměti obrazu pro vyhodnocení jednoho slabého klasifikátoru.



Obrázek 3.13: Hardwarový detektor objektů využívající LRD obrazové příznaky. Převzato z [34].

Hardwarový detektor využívající lokální obrazové příznaky představili Zemčík a Žádník v [34]. Detektor využívá jako slabé klasifikátory LRD obrazové příznaky, které mají na hardwarových detektorech mnohem lepší vlastnosti než Haarovy příznaky. Schéma detektoru je naznačeno na obrázku 3.13. Detektor umožňuje vyhodnocovat až dva obrazové příznaky za jeden takt. Do vnitřní paměti si ukládá jen aktuální výřez obrazu o rozměrech 31×128 obrazových bodů spolu s předvypočítanými podvzorkovanými verzemi obrázku

s měřítkem 1×2, 2×1 a 2×2, což vede k výrazné úspoře paměti na čipu. Na druhou stranu ukládání pouze pruhu obrazu neumožňuje přímou detekci objektů v různém měřítku. Aby bylo možno detekovat objekty ve všech velikostech, je potřeba tuto jednotku doplnit o DSP procesor s velkou pamětí, který by sloužil pro vytváření pyramidy obrazu s různým měřítkem (viz Obrázek 3.1). Další nevýhodou tohoto detektoru je velice složitá síť multiplexorů na výstupu paměti obrazu, jež je způsobeno vyčítáním originálních obrazových bodů spolu s jejich podvzorkovanými verzemi.

Další zajímavé detektory představili například Huang a Valid v [11] nebo Lai a kol. v [14]. Tyto detektory umožňují přímou detekci objektů v různé velikosti. To je dosaženo použitím velké paměti pro uložení celého zpracovávaného obrazu, což umožňuje obraz zmenšovat a na jednotlivých velikostech vyhodnotit klasifikátory pro detekci objektů. Tuto velkou paměť je nutno vytvořit přímo na čipu jednotky, protože použití externí paměti, jež je pomalejší, prodlužuje dobu pro vyhodnocení příznaku. Paměť na čipu je však omezená, a proto není možné tento postup použít například při detekci objektů v obraze s větším rozlišením.

Kapitola 4

Zhodnocení stavu a specifikace zadání

V této kapitole jsou popsány cíle práce. Je stručně zmíněna motivace pro výběr konkrétního tématu, kterým je detekce objektů v obraze pomocí FPGA. Jsou diskutovány dnešní možnosti detekce objektů v obraze na hardwarových jednotkách a jsou vzneseny požadavky na nový navrhovaný systém.

4.1 Motivace

Cílem práce je vybrat vhodný algoritmus zpracování rastrového obrazu a vhodně jej implementovat na programovatelném hradlovém poli. Existuje velké množství algoritmů zpracování rastrového obrazu. Může se jednat o jednoduché algoritmy,jako je například převod barevného obrázku na šedotónový nebo černobílý, detekce hran nebo prahování. Tyto algoritmy jsou však velice jednoduché a výkonově nenáročné, tudíž je možné je velmi dobře zpracovávat na obecném procesoru nebo případně signálovém procesoru. Existují však jiné algoritmy, jejichž výpočet je velmi náročný na výkon. Jedná se především o metody detekce objektů, HDR zpracování obrazu nebo například pokročilé metody segmentace obrazu. Pro tyto algoritmy platí, že jejich výpočet v reálném čase je na hranici nebo i za hranicí výkonnosti dnešních počítačových systémů. Přitom jsou to velmi užitečné algoritmy použitelné v mnoha oblastech lidského života.

Jako vhodný algoritmus pro implementaci na hradlovém poli jsem si vybral detekcí objektů v obraze. Důvodem je jeho možná reálná použitelnost v mnohých oblastech, jako je průmysl a bezpečnost. Další motivací byla jeho vysoká výpočetní náročnost na počítačových systémech, která v dnešní době neumožňuje tvorbu levných a úsporných zařízení s touto funkčností. Cílem bude tedy prozkoumat možnosti detekce objektů na programovatelných hradlových polích, navrhnou úpravy stávajích řešení a implementovat právě levné a úsporné zařízení pro detekci objektů.

4.2 Možnosti detekce objektů v hardware

V kapitole 3 jsou popsány stávají algoritmy pro detekci objektů v obraze. Jsou zde představeny také existující koncepty hardwarových jednotek pro detekci objektů. Tyto jednotky dosahují různých parametrů v oblasti přesnosti detekce, rychlosti zpracování a požadavků

na zdroje. Jsou založeny především na klasifikátorech trénovaných metodou AdaBoost, avšak často používají odlišné obrazové příznaky. Nejčastěji jsou používany Haarovy příznaky. Jejich použití na hardwarových detektorech je však problematické, jak je popsáno v kapitole 3. Důvodem jejich použití je nejspíš neznalost autorů v oblasti počítačového vidění, neboť jsou vždy implementovány původní klasifikátory vytvořené Viala a Jonesem nebo klasifikátory natrénované pomocí vestavěných funkcí knihovny OpenCV. Autoři neřeší možnost použití jiných příznaků nebo úpravu trénovacího procesu na míru pro hardwarové detektory. Existuje ale i několik konceptů, které nevyužívají Haarovy příznaky, ale například lokální obrazové příznaky, jež jsou mnohem vhodnější. U těchto konceptů se ale objevil problém s vyčítáním dat z paměti pro uložení obrazu. Proto bude muset být v návrhu nového detektoru řešen.

Dalším problémem dnešních detektorů je řešení efektivní detekce objektů různé velikosti. Existuje opět několik konceptů. Jednou z možností je využití dalších součástek, jako je signálový procesor nebo externí paměť. Jinou možností je použití opravdu velkého čipu, na kterém je možné realizovat dostatečně velkou vnitřní paměť pro uložení celého obrazu. Proto budou v návrhu detektoru představeny nové postupy pro detekci objektů různé velikosti, které jsou efektivnější než současné řešení.

Důležitá je také otazka výkonnosti celého systému. Nejlepší současné hardwarové detektory jsou schopny vyhodnotit i více než 100 snímků za sekundu při rozlišení obrazu 640×480 obrazových bodů. Proto se zdá, že další růst výkonosti neni potřebný. Vyšší výkonnost detektoru je však nutná například pro práci s obrazem na vysokém rozlišení, jako je HD video. Vyšší výkonost také umožňuje detekovat současně více tříd objektů zájmů v obraze. Současné detektory nepoužívají žádné pokročilé metody pro optimalizaci detekce, jako je například využití sousedních pozic nebo potlačení nemaximálních pozic. Proto budou pro zvýšení výkosti systému tyto metody zohledněny.

4.3 Specifikace zadání

Navrhovaný detektor bude pracovat jako komplexní jednotka pro detekci objektů. Bude tedy schopen určit, zda se v daném obraze nacházejí hledané objekty zájmu a případně určit jejich polohu. Detektor umožní použití i jako rychlá předzpracovací jednotka omezením počtu počtu zpracovávaných příznaků.

Koncepčně bude vycházet z detektoru autorů Zemčíka a Žádníka [34], ale bude řešit jeho problémy s vyčítáním dat z paměti a možností detekce objektů v různé velikosti. Výhoda jejich konceptu je především v použití lokálních obrazových příznaků a ukládání pouze úzkého pruhu obrazu, se kterým se aktuálně pracuje.

Detektor bude využívat klasifikátor natrenovaný pomocí algoritmu WaldBoost. Tento klasifikátor je mnohem výkonější než klasický klasifikátor trénovaný pomocí algoritmu AdaBoost. Důvodem je možnost zamítnutí pozice při vyhodnocení jen několika příznaků. Algoritmus WaldBoost je také použít z důvodu podpory v dostupném nástroji pro trénování klasifikátorů. Výhodou tohoto nástroje je možnost přispůsobit proces trénování na míru použitému detektoru, a to nastavením používané šířky slova pro alfa tabulky a prahy. Tato vlastnost umožňuje tvorbu klasifikátorů, jejichž koeficienty jsou paměťově nenáročné a současně si zachovávají dobrou přesnost. Pokud by totiž byly použity standardní koeficienty, které jsou většinou uloženy pomocí datového typu float, zabíraly by v paměti mnoho místa. Případné kvantizace těchto koeficientů na menší šířku slova by znamenala zmenšení přesnosti výsledného klasifikátoru. Pokud se ale použije omezení příslušné šířky slova už v průběhu trénování, příliš to nezhorší kvalitu výsledného klasifikátoru.

Kapitola 5

Návrh hardwarového detektoru objektů

V této kapitole je podrobně popsán návrh detektoru objektů ve videu. Je vycházeno z teoretického rozboru představeného v kapitolách 2 a 3 a také z požadavků na systém diskutovaných v kapitole 4.

Na začátku kapitoly je navrženo zapojení detektoru do systému a specifikována příslušná hardwarová architektura, pro kterou bude návrh probíhat. Dále je odhadnuta potřebná výkonnost systému, na jejímž základě je proveden návrh výsledného systému.V kapitole jsou navrhovány nové možnosti detekce objektů různé velikosti a jsou také rozebrány možnosti zvýšení výkonu detektoru. Na konci kapitoly je představena paralelní architektura detektoru.

Návrh bude probíhat pro konkrétní rodinu programovatelných hradlových polí. Není totiž možné použít standardní abstraktní návrh jako například při vývoji software. Důvodem jsou naprosto jiné možnosti konkrétních rodin obvodů a také možnost výrazných optimalizací, které plynou z přizpůsobení návrhu přímo na konkrétní rodinu obvodů.

5.1 Zapojení detektoru do systému

Cílovou platformou je rodina obvodů Spartan6 od společnosti Xilinx. Důvodem je možnost testování na dostupném vývojovém kitu s čipem Spartan6X45T.

Na vývojové kartě je vytvořeno rozhraní pro připojení k počítači přes PCIe1x1, jež je popsáno v [16]. Je vytvořena základní funkcionalita pro přenost dat mezi počítačem a vývojovým kitem a zpět. Je dostupná možnost využití DMA kanálu pro rychlý přenos dat a také vyvolání přerušení na vývojovém kitu, jež je možné zachytit v obslužném programu na počítači. Komunikace přes DMA kanál probíhá bez asistence procesoru počítače mezi operační paměti počítače a dynamickou paměti typu DDR3, která je umístěna na vývojové kartě. Detektor tedy může komunikovat s počítačem přes tuto dynamickou paměť. K dispozici je i sada registrů, které mohou sloužit pro řízení obvodu na vývojové kartě.

Dále je připraveno univerzální rozhraní pro připojení průmyslové videokamery přes síťové rozhraní LAN na UDP/IP protokolu. Jedná se o černobílou videokameru M621 od společnosti Camea. Videokamera produkuje proud 60 snímku za sekundu o rozlišení 752×478 obrazových bodů s rozsahem 8 bitů na pixel. Na straně rozhraní videokamery je jen malá vyrovnávací paměť, proto musí být detektor dostatečně rychlý, aby stihl vyhod-

notit všechny pozice v obraze dříve než budou přepsány.

Obrázek 5.1: Zapojení detektoru objektů v rámci celého systému.

Navrhovaný detektor by měl pracovat jako komplexní jednotka pro detekci objektů ve videu. Úkolem detektoru je zpracovávat data z videokamery a vyhledávat v nich hledané objekty zájmu. Výsledky detekce jsou dále poslány na připojený počítač pomocí PCIe, jak je znázorněno na obrázku 5.1. V budoucnu se předpokládá také připojení monitoru pomocí rozhraní DVI, které je na vývojovém kitu také dostupné.

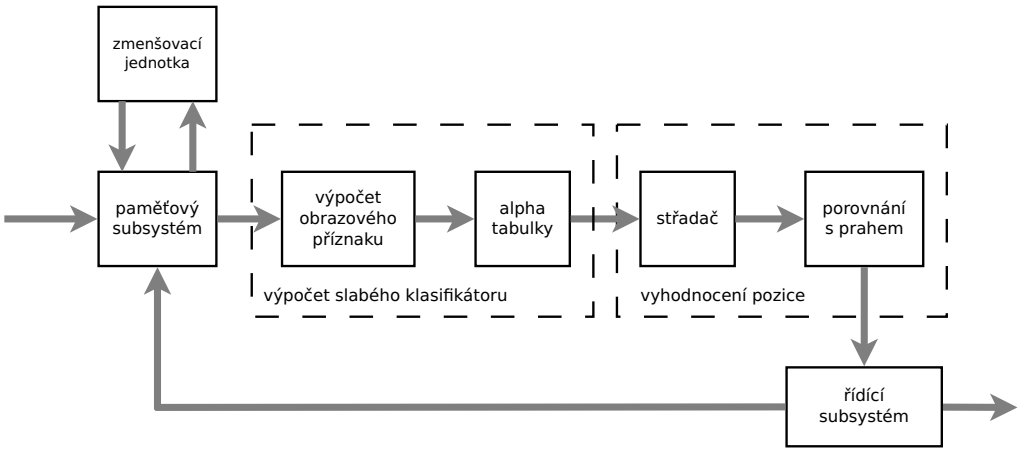
5.2 Rozdělení na subsystémy

Pro zjednodušení návrhu, implementace a testovatelnosti je detektor rozdělen do několika propojených podsystémů. Při rozdělení je vycházeno ze systému navrženého v [34]. Jednotlivé subsystémy jsou:

- *paměťový subsystém*
- *subsystém pro výpočet slabého klasifikátoru*
- *subsystém pro vyhodnocení pozice*
- *řídící subsystém*

Subsystémy jsou vzájemně propojeny, jak je ukázáno na obrázku 5.2. Na vstupu je paměťový subsystém, který slouží pro ukládání pravě zpracovávané části obrazu. Paměťový subsystém slouží i jako vyrovnávací paměť pro připojení videokamery. To je zajištěno tím, že jeho kapacita je o několik řádků větší než je minimální potřebná velikost. K paměťovému subsystému je připojena jednotka pro změnu měřítka. Pracuje s daty v tomto subsystému a vytváří v něm zmenšené verze původního obrazu. K paměťovému subsystému je připojena výkonná jednotka pro výpočet slabého klasifikátoru. Obsahuje jednotku pro výpočet obrazového příznaku a sadu příslušných tabulek s váhou α. Její výsledek je zpracováván subsystémem pro vyhodnocení pozice, který rozhodne, zda se na dané pozici nenachází hledaný objekt. V případě kdy nemůže rozhodnou, inicializuje vyhodnocení dalšího slabého klasifikátoru pro příslušnou pozici. Pro rozhodnutí o pozitivní detekci objektu jsou vyhodnoceny všechny dostupné slabé klasifikátory. Řídící subsystém obsahuje logiku pro řízení běhu detektoru. Jedná se o sekvenční automat, který je řiditelný mikroprogramem. Použití mikroprogramu umožňuje snadnou změnu klasifikátoru za účelem detekce jiných objektů zájmu.

Výsledný detektor pracuje nad obrazem, postupně vyhodnocuje slabé klasifikátory a jejich akumulovaný výsledek porovnává s prahem, čímž rozhoduje detekci objektů na dané pozici. Toto vyhodnocení je provedeno na všech pozicích obrazu.



Obrázek 5.2: Rozdělení detektoru do subsystémů.

Odhad výkonosti

Důležitá je výkonost celého systému. Proto je nutné provést odhad minimální výkonosti. Ta je daná počtem zpracovávaných bodů a průměrným počtem vyhodnocených příznaků na jeden bod podle vzorce 5.2. Počet zpracovávaných bodu je dán především rozlišením okna obrazu, počtem snímků za sekundu a počtem zmenšených verzí obrazu. Průměrný počet obrazových příznaků na jednu pozici je možno zjistit z výsledků trénování.

$$image_{position} = \sum_{i=0}^{\log_k scale_{max}} \frac{(image_x - window_x) \cdot (image_y - window_y)}{k^{2i}} \tag{5.1}$$

$$features_{peersecond} = image_{position} \cdot image_{peersecond} \cdot features_{average} \tag{5.2}$$

Pro navrhovaný systém je použit vstup z videokamery o rozlišení 752×478 obrazových bodů při 60 snímcích za sekundu. Rozměry skenovacího okna jsou 24×24. Maximální zmenšení je 16 a poměr mezi jednotlivými zmenšenými verzemi 1,2. Průměrný počet obrazových příznaků na jednu pozici určený z trénování je 5. Pro tuto konfiguraci vychází minimální odhad výkonnosti 325 milionů obrazových příznaků za sekundu. Na použitém FPGA čipu, který může běžet na maximální frekvenci 200 MHz, je tedy nutné vyhodnotit průměrně více než 1,6 obrazových příznaků za takt. Z toho vyplývá, že bude nutné použít system s více jednotkami pro výpočet obrazového příznaku nebo bude nutné použít některé metody pro urychlení detekce objektů, jako je využití sousedních pozic nebo potlačení nemaximálních pozic.

Důležitý je také odhad požadovaného výkonu pro jednotku plnící paměťový subsystém daty z kamery. Rozhraní kamery posílá data po čtyřech hodnotách za takt s přibližně rovnoměrným rozložením v čase. Na straně rozhraní je také malá vyrovnávací paměť pro uložení jednoho řádku, která případně pokryje prostoje detektoru. Požadovaný počet zápisu je dán rozlišením okna obrazu a počtem snímků za sekundu, jak ukazuje vzorec 5.3.

$$write_{peersecond} = image_x \cdot image_y \cdot image_{peersecond} \tag{5.3}$$

Pro parametry navrhovaného systému vychází přibližně 5,5 milionů zápisů za sekundu. Tato hodnota neni v porovnání s frekvencí FPGA kritická, a proto neni nutné zapis zvlášť optimalizovat.

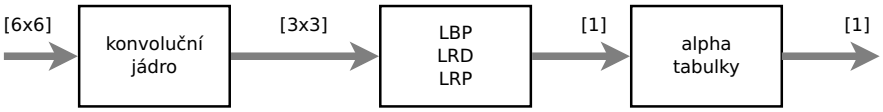
Odhad výkonnosti jednotky pro zmenšování obrazu bude závislý na postupu detekce objektů různé velikosti, proto bude popsán dále.

5.3 Výpočet slabých klasifikátorů

Detekce objektů pomocí klasifikátorů je založena na principu skládání slabých klasifikátorů. Pro detekci objektů v obraze se používají jako slabé klasifikátory obrazové příznaky. Nejrozšířenějšími jsou Haarovy příznaky, které však nejsou vhodné pro použití na hardwarových detektorech. Další možností jsou lokalní obrazové příznaky, které jsou vhodné pro toto použití. Z toho důvodu budou lokální obrazové příznaky použity jako slabé klasifikátory v navrhovaném detektoru.

Detektor využívá lokální obrazové příznaky typu LBP a LRF. Pro zvýšení počtu příznaků nad skenovacím oknem, a tím ke zmenšení chyby detekce, je použit postup popsaný v [20]. Vyhodnocení příznaků tedy probíhá nad odezvou konvolučního jádra s originálním obrazem. Rozměry tohoto jádra jsou 1×1, 1×2, 2×1 a 2×2.

Výpočet odezev konvolučních jáder s originálním obrazem může být prováděn před samotnou detekcí. Jednotlivé odezvy jsou pak uloženy v paměťovém subsystému. Tento přístup je velice efektivní, protože počet výpočtů odezvy je závislý na součinu počtu bodů obrazu a počtu konvolučních jader. Nevýhodou je však větší paměťová náročnost a také neregulární přístup do paměti, který vede u hardwarových detektorů k vytvoření složité přepínací sítě na jejím výstupu, jak je ukázáno například v [34]. Z toho důvodu bude probíhat výpočet odezvy vždy před samotným vyhodnocením příznaku. Zmenší se tak paměťová náročnost a také dojde k zjednodušení vyčítání dat z paměti. Počet výpočtů odezvy se zvětší, ale je možné ho efektivně skrýt do zřetězené linky detektoru. Nevýhodou tohoto přístupu je ale nutnost vyčítání větších bloků dat na jeden obrazový příznak. Nejvíce pro konvoluční jádro 2×2 , kde je při použití obrazového příznaku o rozměru 3×3 nutné vyčítat z paměti blok 6×6 obrazových bodů místo původních 3×3. Při vyčítání se však vždy jedná o kompaktní pravoúhlý blok paměti, umožňující snadný regulární přístup do paměti.



Obrázek 5.3: Struktura jednotky pro výpočet obrazového příznaků.

Z důvodu uniformního přístupu načítá jednotka pro výpočet obrazového příznaku z paměti obrazu vždy blok o rozměru 6×6 obrazových bodů. Tato data jsou poté zpracována filtrem s požadovaným konvolučním jádrem a následně poslána do samotného detektoru, jak je ukázáno na obrázku 5.3.

Obrazové příznaky

Existuje několik variant obrazových příznaků založených na LBP. Tyto varianty se liší především rozměrem, který je dán počtem porovnávaných bodů nebo také v následné úpravě jako je použito například u rotovaných LBP příznaků. V detektoru jsou použity standardní LBP příznaky o rozměru 3×3. Důvodem je především dobrá přesnost a snadná implementace. Nevýhodou je naopak poměrně velká tabulka s váhami α, která pro každy tento příznak obsahuje 256 hodnot.

Experimentálně jsou také použity obrazové příznaky o rozměru 2×2. Chování těchto příznaků není přiliš ověřeno. Teoreticky by měly dosahovat horší přesnosti. Na druhou stranu potřebuji jen malou tabulku s vahami α pro každý příznak s 16 hodnotami.

Samotný výpočet LBP příznaků je jednoduchý a spočívá v několika porovnání příslušných hodnot obrazových bodů. Výsledky těchto porovnání jsou poté jako maska uloženy do hodnoty příznaku.

Obrazové příznaky typu LRF poskytují srovnatelnou přesnost s LBP příznaky. Jejich výhodou je použití menší tabulky s váhami α. V detektoru jsou použity obě varianty příznaků, a to LRP a LRD. Tyto příznaky pracují s blokem dat o rozměru 3×3 bodů. Pro příznaky LRP je potřebná tabulka s 81 hodnotami α a pro příznaky LRD s 17 hodnotami α.

Výpočet LRF příznaků je založen na určení pořadí hodnot v dané oblasti obrazu. K výpočtu hodnoty příznaků je zapotřebí určit pořadí dvou obrazových bodů. To je provedeno pomocí porovnání každého z těchto bodů s ostatními body v oblasti, protože provádět kompletní setřídění celé této oblasti je výkonově velmi náročné. Ze znalosti pořadí dvou příslušných bodů je určen LRD příznak jejich odečtením a LRP jejich konkatenací.

Tabulky s váhou klasifikátoru

Pro určení hodnoty slabého klasifikátoru je potřeba zjistit jeho váhu z příslušné sady tabulek. U lokálních obrazových příznaků platí, že počet vah slabého klasifikátoru je dán počtem kombinací, kterých může obrazový příznak nabývat. Potřebná velikosti tabulky pro příslušné obrazové příznaky je vypsána v tabulce 5.1

obrazový příznak	LBP 3×3	LBP 2×2	LRD 3×3	LRP 3×3
velikost tabulky	256	16	17	81

Tabulka 5.1: Potřebná velikost tabulky s váhou α pro jednotlivé příznaky.

Potřebná kapacita pro uložení tabulek s váhou α je dána součinem počtu příznaků silného klasifikátoru, velikostí tabulky pro uložení jednoho příznaku a také datovou šířkou jednotlivých hodnot v tabulkách. Počet příznaků je závislý na požadované přesnosti detektoru, neboť s počtem příznaků roste i teoretická přesnost detekce. Počet příznaků je možno omezit, potom se však detektor bude chovat jako rychlá předzpracovací jednotka a výsledky bude nutné dále zpracovat jiným systémem. Datovou šířku jednotlivých hodnot v tabulce je možno omezit. Při výpočtu na procesorových systémech se nejčastěji používají 32 bitové neceločíselné typy. Pokud je zohledněna požadovaná šířka dat při trénování, je možné použít datové typy s výrazně menší šířkou dat, jak je naznačeno v kapitole 4.

5.4 Subsystém pro vyhodnocení pozice

Úkolem tohoto subsystému je určit, zda se na právě vyhodnocované pozici nachází hledaný objekt zájmu. Pro detekci využívá silný klasifikátor trénovaný metodou WaldBoost. To umožňuje zamítnout danou pozici často ještě před vyhodnocením celého klasifikátoru.

Subsystém ukládá součet vah vyhodnocených obrazových příznaků pro aktuální pozici. Tento součet je porovnáván s prahem, který je pro každý klasifikátor určen při trénování. Pokud je součet menší než aktuální práh, je pozice zamítnuta. Pokud je ale větší, dochází k vyhodnocení dalšího slabého klasifikátoru pro tuto pozici, dokud nejsou takto vyhodnoceny všechny. Pokud se pro danou pozici vyhodnotí všechny slabé klasifikátory a součty jsou vždy větší než příslušný práh, je na dané pozici detekován hledaný objekt zájmu.

Pro vyhodnocení jednoho slabého klasifikátoru je zapotřebí provést mnoho paměťových a výpočetních úkonů. Jedná se minimálně o načtení instrukce z paměti mikroprogramu, načtení příslušného bloku dat z paměti obrazu, vyhodnocení obrazového příznaku, načtení váhy slabého klasifikátoru, akumulace váhy a její porovnání s prahem. Provedení všech těchto operací bude jistě trvat několik hodinových taktů. Vhodné rozdělení detektoru na části s porovnatelnou časovou složitostí umožní zřetězené zpracování úlohy pro vyhodnocení slabého klasifikátoru. Pro efektivní využití této zřetězené linky je potřeba současně vyhodnocovat několik slabých klasifikátorů současně. Jejich počet je dán hloubkou linky. Pokud by se jednalo o slabé klasifikátory příslušející jedné pozici, bylo by nutné při předčasném zamítnutí zahodit provedené výsledky ve zřetězené lince, což je neefektivní. Jinou možností je vyhodnocovat slabé klasifikátory současně na více rozdílných pozicích obrazu. Toto řešení je velmi efektivní. Problémem je však to, že se může počet vyhodnocených slabých klasifikátorů na každé pozici lišit, z čehož vyplývá složitější řízení pro výběr další pozice pro vyhodnocení.

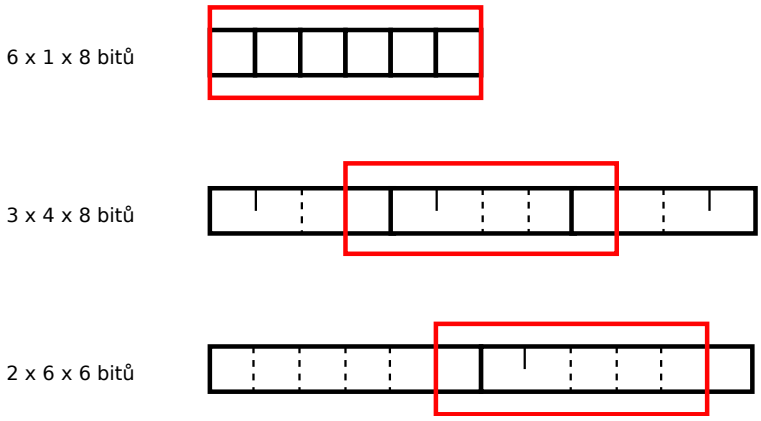
5.5 Paměťový subsystém

Úkolem paměťového subsystému je uchovávat aktuální oblast zpracovávaného obrazu a umožnit rychlé vyčítání bloků obrazu z ní. Vyčítání dat z paměti probíhá po blocích o velikosti 6×6 obrazových bodů, což vyplývá z návrhu subsystému pro výpočet slabého klasifikátoru. Jedná se o velké množství dat, což je nutné zohlednit př návrhu.

Rodina programovatelných hradlových polí Spartan6 umožňuje vytvořit interní paměť pomocí obecných nebo posuvných registrů vestavěných v Slice blocích nebo pomocí distribuované paměti BRAM. Použití registrů neni možné z důvodu jejich malé kapacity, proto je paměťový subsystém tvořen pomocí paměti BRAM. Těchto pamětí je k dispozici na čipu vývojového kitu celkem 116. V konfiguraci těchto pamětí je možno nastavit požadovanou šířku datového slova, a to 1, 9, 18 nebo 36 bitů. Data z kamery mají šířku datového slova 8 bitů. Pokud by byla pro uložení obrazu použitá konfigurace s šířkou slova 9 bitů, která je nejbližší, bylo by nutné pro efektivní vyčítání hodnot bloku 6×6 z paměti použít současně 36 BRAM, jak je ukázáno na obrázku 5.5.

Další možností je použít šířku slova 36 a ukládat současně 4 hodnoty. Pro vyčítání požadovaného bloku by bylo zapotřebí 18 BRAM. Jinou možností je oříznout dva nejméně významné bity dat z kamery a uložit do 36 bitového slova 6 hodnot současně. Chyba na datech způsobená oříznutím bude $1,5\%$, což je na hranici šumu a nemělo by proto ovlivnit přesnost detekce. S touto konfigurací je pro vyčítání požadovaného bloku zapotřebí jen 12 BRAM. Úspora pamětí používaných pro vyčítání bloku je důležitá z důvodu následné

paraelizace celého detektoru, proto bude tato poslední navrhovaná architektura použita. Důležité je vhodné uspořádání dat v paměti. Aby bylo umožněno vyčítání dat v bloku, musejí se paměti BRAM na řádku vhodně střídat.



Obrázek 5.4: Ukázka vyčítání jednoho řádku šesti hodnot z různé konfigurace pamětí. Hodnota uvádí aktuální konfiguraci, první číslo počet BRAM, druhé počet hodnot v BRAM a třetí šířku datového slova.

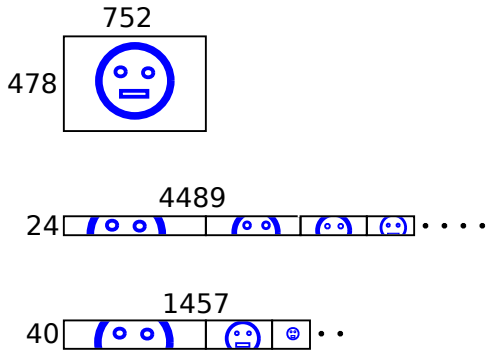
Pro detekci objektů jedné velikosti je kapacita pamětí paměťového subsystému dána velikostí aktuálně ukládané oblasti obrazu. Tato oblast je dána především povahou rozhraní vstupních dat. Videokamera produkuje proud dat, které zasílá vedle sebe po jednotlivých obrazových bodech, řádek po řádku. To znamená že je načten první řádek, po něm druhý atd. Poté již kamera znovu nepošle data, která jsou v aktuálním obrázku na prvním řádku. Pokud s nimi chce detektor pracovat, musí si je uložit. To znamená, že pokud je potřeba pracovat s daty na více řádcích, musejí být celé uloženy v paměti.

Při detekci objektů je minimální počet řádků, s kterými se aktuálně pracuje, dán rozměry skenovacího okna klasifikátoru. Detektor používá skenovací okno o rozměru 24×24 obrazových bodů. Z toho vyplývá, že minimální velikost paměti v bitech bude součin šířky obrazu, velikosti skenovacího okna a datové šířky pro uložení obrazového bodu. Pro navrhovaný systém s organizací $752 \times 24 \times 6$ bitů je to 106 Kb paměti, která se vejde do 6 BRAM pamětí. Avšak z důvodu vyčítání je potřeba minimálně 12 pamětí BRAM, které umožní ukládání až 48 řádků dat v paměti. Tato nadbytečná paměť však neni na škodu, protože je použita jako vyrovnávací paměť při práci s kamerou.

Detekce objektů v různém měřítku

Detekce objektů různé velikosti se v dnešní době řeší na hardwarových detektorech pomocí externích součástek, jako jsou signálové procesory, které vytvářejí zmenšená verze původního obrazu. Další možností je ukládání celého obrazu do vnitřní paměti na čipu nebo externí paměti mimo čip, která umožňuje následné zmenšování. Použití externího procesoru nebo paměti zdražuje cenu výsledného detektoru a výrazně komplikuje návrh desky plošných spojů. Vytvoření interní paměti uvnitř čipu si vynucuje použití opravdu velkých FPGA čipů, které jsou také velice drahé. Navíc například při práci s obrazem s vysokým rozlišením neni možné ani na dnešních největších čipech potřebnou paměť vytvořit.

Pro návrh nového detektoru platí, že není možné použít žádný externí signálový procesor. Důvodem je požadovaná funkčnost na vývojovém kitu, který žádný takový procesor neobsahuje. Použití externí paměti je možné, protože je na vývojovém kitu dostupná. Rozhodl jsem se ji ale nepoužít, a to z důvodu dalšího možného použití detektoru na jiných platformách, jejíž cenu by tato paměť zvětšovala. Vytvoření dostatečně velké interní paměti na čipu neni také možné, protože k jejímu vytvoření je zapotřebí 118 pamětí BRAM a čip obsažený na kitu jich má pouze 116.



Obrázek 5.5: Ukázka uložení dat v paměti pro detekci objektů různé velikosti. Nahoře je konfigurace ukládající celý obraz, uprostřed při použití pruhů s zmenšenými verzemi a dole s pruhem se zmenšováním na polovinu.

Proto jsem navrhl nový přístup pro detekci objektů různé velikosti na hardwarových detektorech. Vychází z postupu pro detekci objektů jedné velikosti, kdy se ukládala do paměti jen oblast s kterou se aktuálně pracuje. Vedle ní se ale nově ukládají další pruhy, které obsahují zmenšené verze původního obrázku. První zmenšená verze se generuje z původního obrázku a následující vždy z předchozí verze. Pro zmenšování se používá stejný poměr jako při zmenšování celého obrazu. Velikost paměti potřebná pro uložení aktuální oblasti se zvětší. Počet řádků zůstane stejný, avšak šířka potřebné paměti výrazně vzroste. Aktuální šířka bude dána velikostí původního obrazu, poměrem zmenšení mezi dvěma verzemi obrazu a maximálním požadovaným zmenšením podle vzorce 5.4. Pro navrhovaný systém se tedy šířka dat ukládaných v paměti zvětší na 4484 sloupců. Pro uložení této paměti bude zapotřebí celkem 36 pamětí BRAM. Tento navrhovaný postup dovoluje detekovat objekty různé velikosti bez nutnosti uložení celého obrazu do paměti.

$$mem_{width} = \sum_{i=0}^{\log_k scale_{max}} \frac{image_x}{k^{2i}} \tag{5.4}$$

Při implementaci tohoto navrhovaného postupu se zjistilo, že jednotka, která provádí samotné zmenšování spotřebovává skoro 70% hradel paměťového subsystému. Je použita jednotka pro zmenšení v standardním poměru 1.2, který se pro detekci objektů různé velikosti obecně používá. Navíc pro potřebné zmenšení bloku dat o velikosti 6×6 na 5×5 obrazových bodů, bylo zapotřebí velmi mnoho hodinových taktů pro výpočet, což bylo dáno především nezarovnaným přístupem do paměti. Tento nezarovnaný přístup byl způsoben především ukládáním pěti hodnot do paměťové buňky s šesti hodnotami.

Z toho důvodu byla navržena další úprava. Její princip je podobný jako u předchozího

řešení, ale do paměti si ukládá pruhy obrazu zmenšené vždy jen na polovinu. Toto zmenšení je velice dobře realizovatelné v hardware a jeho použití vede k zarovnanému přístupu do paměti. Detekce ostatních velikostí mezi dvěma polovičními jsou detekovány pomocí změny velikosti skenovacího okna. Tuto změnu je možné efektivně realizovat pomocí tabulky, která upravuje adresování v paměťovém subsystému nebo pomocí nových klasifikátorů natrénovaných pro jinou velikost okna. Velikost paměti potřebná pro uložení aktuální oblastí se oproti předchozímu návrhu zmenší. Důvodem je snížení počtu sloupců, jejíž počet lze vypočítat podle vzorce 5.5. Počet řádků paměti se zvětší, což je způsobeno použitím detekovacího okna s větší velikostí. Pro navrhovaný systém se tedy počet sloupců paměti zmenší na 1457 a počet řádků vzroste na 40. Pro uložení této paměti bude zapotřebí celkem 20 pamětí BRAM.

$$mem_{width} = \sum_{i=0}^{\log_{0.5} scale_{max}} \frac{image_x}{0.5^{2i}} \tag{5.5}$$

5.6 Akcelerace detekce

Zvýšení výkonosti je důležité z důvodu správné činosti detektoru. Z odhadu výkonnosti vyplývá že navrhovaný detektor musí zpracovat více než 1.6 obrazových příznaků za takt. Při reálném nasazení bude toto číslo ještě větší z důvodu zvýšení počtu vyhodnocovaných příznaku na pozicích obsahujících hledaný objekt zájmu. Případné další výrazné zvýšení výkonnosti by umožňovalo připojení další kamery a detekci objektů nad obrazem v ní. Další možností jak využít přebytečný výkon, je také možnost detekce více tříd objektů v obraze. Pro zvýšení výkonnosti detekce objektů v obraze se na procesorových výpočetních systémech používá několik algoritmů, jež byly představeny v kapitole 3. V navrhovaném detektoru je použit algoritmus využívající sousedních pozic. Detektor je rovněž vybaven vstupem z předzpracovací jednotky.

Využití sousedních pozic

Algoritmus využívající sousední pozice slouží pro zrychlení detekce negativně detekovaných vzorů. Na základě vyhodnocení obrazových příznaků na jedné pozici je schopen rozhodnout o negativní detekci na sousedních pozicích.

Algoritmus pro výpočet využívá hodnotu obrazového příznaku na aktuální pozici. Pro vyhodnocení slabého klasifikátoru na sousedních pozicích však používá vlastní sadu tabulek s váhou α a příslušné prahy pro každý sousední bod. Pro použití v hardwarových detektorech je tedy zapotřebí upravit jejich vyčítání z paměti. V detektoru je použito vyhodnocování 7 okolních pozic. Proto je nutné vyčítat z paměti celkem osm hodnot s váhou α a osm prahů pro vyhodnocení jednoho slabého klasifikátoru. Dále je nutné upravit subsystém pro vyhodnocení pozice, aby umožňoval provádět součty pro každý okolní bod samostatně. Pro potřeby algoritmu je vytvořena kruhová bitmapa, jejíž šířka odpovídá šířce obrazu a výška třem řádkům. Tato bitmapa zamezuje znovu vyhodnocovat pozice, které byly negativně detekovány s využitím algoritmu sousedních pozic.

Použití předzpracovací jednotky

Další možností, jak urychlit práci detektoru, je použití rychlé předzpracovací jednotky, která by byla umístěna na čipu v kameře, jež se stará o její ovládání a posílání dat po síťovém

rozhraní. Na dodané kameře je možnost takovou funkčnost implementovat. Zabývá se jí kolega Vít Široký. Úkolem předzpracovací jednotky je rychle vyhodnotit prvních pár příznaků a určit vhodné body, které jsou pak dále prozkoumávány. Použití této jednotky by mohlo výrazně zvýšit výkon detektoru.

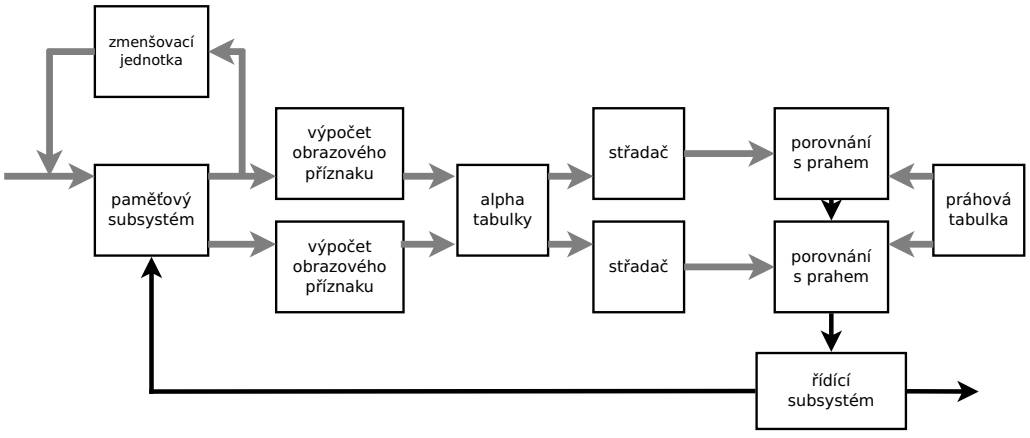
Důležitá je také otázka, zda použití předzpracovací jednotky nepovede ke zhoršení efektivity detektoru. V případě, že by počet pozic vhodných k dalšímu prozkoumání určený předzpracovací jednotkou byl malý, mohlo by docházet v detektoru ke stavu, kdy byl počet taktů k načítání obrazu větší než počet taktů k detekci objektů nad vybranými pozicemi. V tomto případě by se hardwarový detektor stal neefektivním a bylo by lepší místo něj použít obecný procesorový systém, který by vykonával ještě jiné ukoly. Pro ověření této hypotézy je navrhovaný detektor vybaven vstupem z předzpracovací jednotky.

Paralelizace systému

Jinou možností, jak zvýšit výkonnost detektoru, je paralelizace některých jeho částí. Ta by umožňovala souběžný výpočet více obrazových příznaků za jeden takt.

Při implementaci první verze detektoru bylo zjištěno, že subsystémy pro výpočet slabého klasifikátoru a vyhodnocení pozice zabírají velice málo zdrojů na čipu. Proto je možné je velice snadno duplikovat a použít pro paralelní výpočet. Problém je však s jednotkami, které pracují s pamětí BRAM. Té je na čipu nedostatek a jednoduchá duplikace těchto jednotek tak z důvodu kapacity neni možná. Proto je nutné upravit návrh jednotky pro ukládání aktuálně zpracovávané části obrazu a také paměti s váhou klasifikátoru α a příslušných prahů.

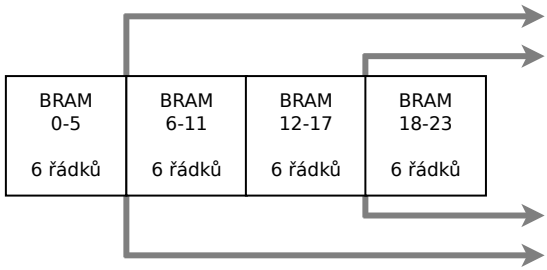
Použité paměti BRAM jsou dvouportové. To znamená, že umožňují dva nezávislé přístupy k paměti. Minimálně jeden z těchto portů musí být použit pro plnění paměti a pro připojení jednotky pro změnu měřítka. Tyto operace zaměstnávají jeden port paměti přibližně na dobu 15 procent z celkového času. Zbylých 85 procent času je tedy možné použít tento port pro výpočet slabých klasifikátorů. Druhý port paměti je trvale použit pro výpočet slabých klasifikátorů. Paměti pro uložení tabulek s váhou α a prahů jsou také dvouportové.



Obrázek 5.6: Paraelní architektura detektoru pro vyhodnocení více slabých klasifikátorů za jeden takt.

Jejich výhodou je, že slouží jen pro vyčítání, takže je možné jednoduše oba porty použít pro vyhodnocení klasifikátorů. Pokud je tedy použít tento navrhovaný postup a současně jsou použity dvě jednotky pro vyhodnocení slabého klasifikátoru a dvě jednotky pro vyhodnocení obrazového příznaku, jak je znázorněno na obrázku 5.6, je možné vypočítat pruměrně 1.85 obrazových příznaků za jeden takt.

Další možností, jak zvýšit počet portů paměti a tím zvýšit i možnosti paralelizace celého systému, je úprava jejího rozložení. Při použití 12 pamětí BRAM je možno vyčítat dva nezávislé bloky o velikosti 6×6, jak bylo ukázáno dříve. Pokud je však použit dvojnásobek pamětí BRAM, tedy 24, je možné data z prvních 12 BRAM duplikovat v druhých. Toto zapojení umožňuje zvýšit počet nezávislých portů na 4. Avšak kapacita paměti z důvodu duplikace zůstává stejná jako při použití 12 BRAM, je tedy efektivně využita jen polovina paměti. Jinou možností je vhodně uspořádat těchto 24 pamětí BRAM, jak je ukázáno na obrázku 5.7. Paměti umístěné v jednom řádku se pravidelně vhodně střídají. To umožňuje vytvořit čtyřportové paměti, které efektivně využívají celou paměť. Nevýhodou tohoto přístupu je závislost jednotlivých portů. Není možné současně vyčítat na čtyřech portech data z prvního sloupce, ale je možné vyčítat data na dvou portech z prvního sloupce a současně na zbylých dvou portech například z dvanáctého. Tento navrhovaný postup umožňuje vytvořit i víceportové paměti, například při použití 36 pamětí BRAM je možné vytvořit šestiportové paměti a při použití 48 BRAM osmiportové paměti. Z důvodu kapacity je tento postup použit v navrhovaném detektoru.



Obrázek 5.7: Úprava rozložení pamětí pro zvýšení jejího počtu portů.

Problém však nastává s řešením závislosti přístupů do paměti. Je možné jej řešit čekáním některých pozic, jejiž vyhodnocení vede ke konfliktu. Tento přístup však neni efektivní. Je možné také navrhnou lepší způsob řízení výběru pozice pro vyhodnocení. Takový algoritmus je však velice složitý a v hardware obtížně realizovatelný. Důvodem je, že počet slabých klasifikátorů, které je nutné vyhodnotit pro detekci, může být na každé pozici jiný. Další možností je využít pokročilé techniky, jež se používají v moderních procesorech. Vhodnou technikou je zpracování instrukcí mimo pořadí. Je možné ji efektivně realizovat pomocí bufferu instrukcí a přídavné logiky, která vybírá instrukce, jež nejsou v konfliktu. Tento poslední přístup je použit v navrhovaném detektoru.

Dalším úzkým hrdlem jsou tabulky s váhou α a prahem. Ty využívají také dvouportové paměti BRAM. K realizaci víceportových tabulek je možné použít techniky, jež jsou posány výše. Pro tabulky s váhou α a prahy platí, že statisticky se bude mnohem častěji přistupovat k tabulkám, které využívá prvních pár slabých klasifikátorů. K dalším tabulkám se bude přistupovat poměrně zřídka. Proto jsou v navrhovanáho detektoru duplikovány jenom ty tabulky, které náleží právě k prvním několika klasifikátorům.

Kapitola 6

Implementace detektoru a jeho testování

Tato kapitola podrobně popisuje realizaci detektoru objektů ve videu, který byl navržen v kapitole 5. Je rozebrána implementace jednotlivých podsystémů a systému jako celku. Jsou vyčísleny implementační vlastnosti systému, jako je požadavek na zdroje a časové charakteristiky detektoru. V závěru kapitoly je také popsáno testování detektoru a jednotlivých podsysémů.

6.1 Popis realizace

Realizace detektoru je rozdělena do modulů, které odpovídají jednotlivým subsystémům popsaných v návrhu systému. Vývoj probíhal v iteracích, a proto existuje několik verzí každého modulu. Stejné moduly lze většinou zaměmit a tím ovlivnit chování celého systému.

Moduly z první verze detektoru umožňují jednoduchou detekci objektů v obraze. Tato verze byla původně plánovaná pro připojení kamery s rozlišením 752×480 obrazových bodů s maximálně 30 snímky za sekundu. Pro vyhodnocení slabých příznaků využívá obrazové příznaky typu LBP, LRD a LRP. Tato verze řešila elegantně problém s vyčítáním dat z paměťového subsystému, avšak neuměla detekovat objekty různé velikosti a nevyužívala žadnou akcelerační metodu pro zrychlení detekce celého obrazu.

Po zjištění skutečných parametrů kamery, která je k dispozici a produkuje 60 snímků za sekundu, byla vytvořena druhá verze, jež umožňuje detekci nad videem z ní. K tomu je využito zpracování dvou slabých klasifikátorů současně pomocí jejich paralelizace. Ve třetí iteraci byl vyvinut detektor, který umožňoval detekci objektů různé velikosti a také mohl využívat vyhodnocení sousedních pozic pro zrychlení detekce obrazu.

Čtvrtá a zároveň poslední verze detektoru využívá masivní paralelismus. Ten je dán použitím paměťového subsystému s osmi porty a obvody pro vyhodnocení slabých klasifikátorů současně na šesti pozicích.

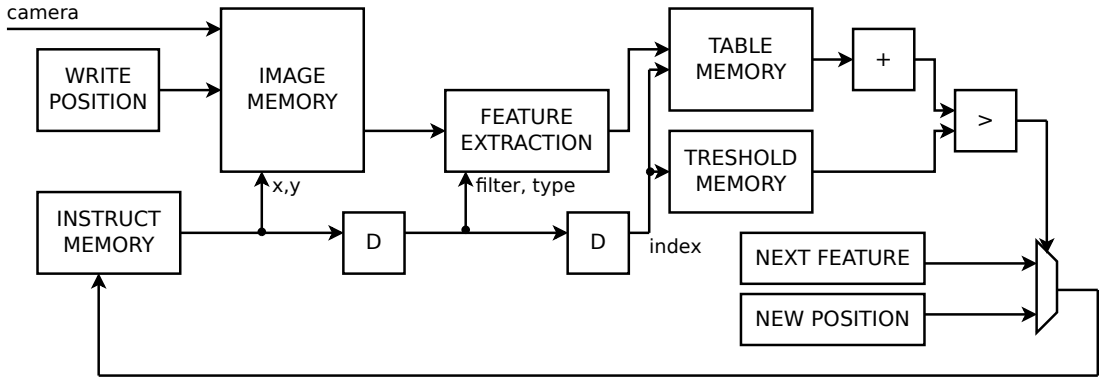
Realizace detektoru byla přizpůsobena na konkrétní FPGA čip Spartan 6 X45T. Systém byl implementován s pomocí jazyka pro posis hardware VHDL. Při jeho vývoji byly využity nástroje dodávané výrobcem čipu firmou Xilinx. Jedná se o vývojové prostředí ISE a nástroj pro simulaci iSim.

Implementace je přizpůsobena k tomu, aby výsledný systém mohl běžet na hodinovém kmitočtu 200 MHz. Současně se snaží efektivně využít dostupné jednotky a snížit tím

množství potřebných zdrojů. Je tedy implementována s omezením na dodržení hodinového kmitočtu a využitím minima zdrojů.

6.2 Architektura detektoru

Celková architektura základní verze detektoru je zobrazena na obrázku 6.1. Ostatní verze se liší pouze upravou některých subsystémů nebo jejich paralelizací. Náklady na zdroje jednotlivých verzí jsou uvedeny v tabulce 6.1.



Obrázek 6.1: Zjednodušené schéma implementovaného detektoru.

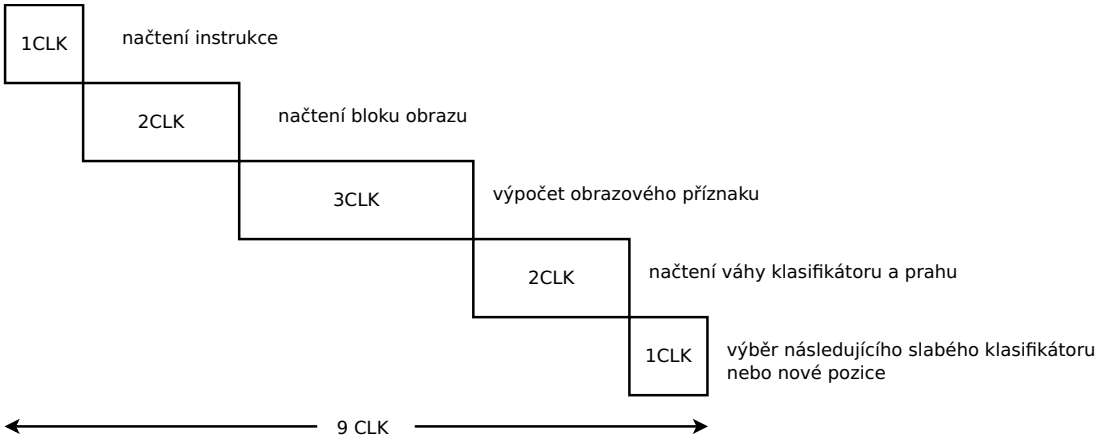
	Frenvence MHz	Slice počet	Slice procent	BRAM počet	BRAM procent	Různá velikost	Využití sousedů	Stupeň parelelizmu
verze 1	203	523	8	31	27	N	N	1
verze 2	203	955	14	43	37	N	N	2
verze 3	203	1064	16	93	80	A	A	2
verze 4	151	2708	40	96	82	A	N	6

Tabulka 6.1: Náklady na zdroje a parametry jednotlivých verzí detektoru.

Řízení detektoru

Implementovaný detektor pracuje jako programovatelný automat a jeho chod je řízen instrukcemi. Ty jsou načítány z paměti instukcí a přes zpožďovací linky rozvedeny k jednotlivým modulům. Pužití zpožďovacích linek je nutné pro správnou funkčnost zřetězené linky. Detektor využívá zřetězenou linku, která má devět částí. Pro efektivní využití této linky tedy zpracovává devět pozic obrazu současně. Počet příznaků za takt se blíží skoro k jedné. To je dáno dobrým poměrem počtu pozic obrazu a času k naplnění linky. Časová charakteristika zřetězené linky je zobrazena na obrázku 6.2.

Detektor využívá instrukce s šířkou slova 36 bitů. Pro jejich uložení je použita jedna paměť BRAM. Maximální počet instrukcí je 1024. Počet instrukcí pro pozitivní vyhod-

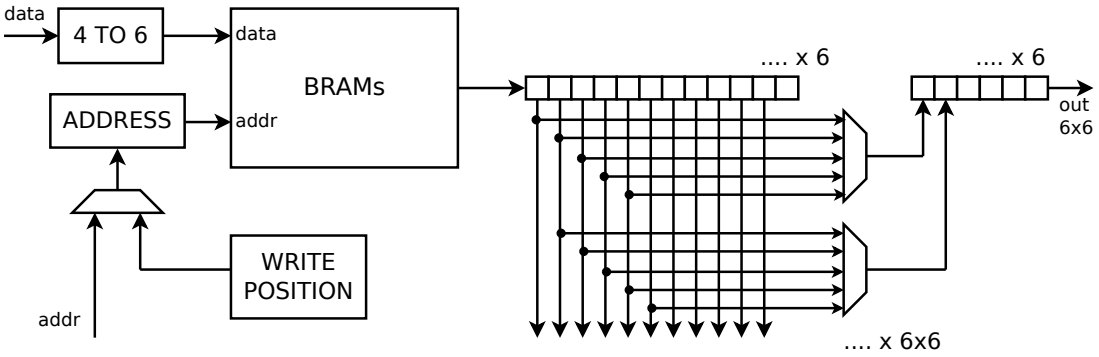


Obrázek 6.2: Časové schéma zřetězené linky detektoru.

nocení pozice je možno nastavit v konfiguračním zdrojovém souboru. Instrukce obsahuje údaje o pozici vyhodnocovaného bloku obrazu v rámci skenovacího okna, identifikaci použitého jádra filtru, parametry obrazového příznaku a měřítko použité pro zmenšení obrazu. Pro případné rozšíření je dostupných ještě deset volných bitů. Detektor efektivně využívá číslo instrukce, které slouží pro načítání příslušných tabulek s váhou α a prahů pro zamítnutí pozice.

Paměť obrazu

Paměť obrazu je implementována tak, aby umožňovala rychlé vyčítání bloků o rozměru 6×6 obrazových bodů. Pro úsporu místa a zjednodušení vyčítání dat ukládá pouze horních šest bitů hodnoty obrazového bodu. Paměť je navenek organizovaná jako 2D, a proto se adresuje dvojicí hodnot x, y. Hodnota x určuje pozici na řádku a y pozici ve sloupci. V paměti není uložen celý obrázek, ale jen jeho aktuální pruh. Z toho důvodu je paměť kruhová v ose y.

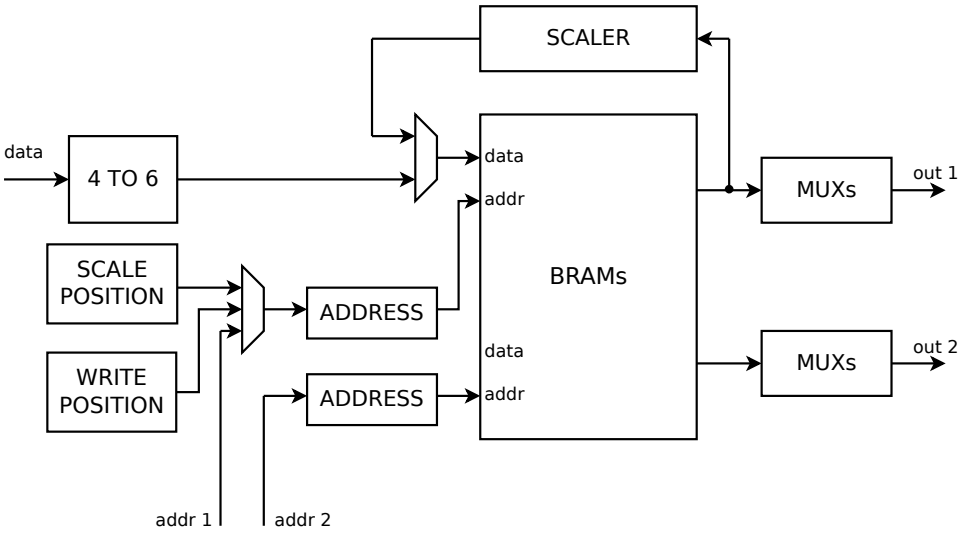


Obrázek 6.3: Zapojení paměti obrazu pro vyčítání bloku 6×6 obrazových bodů.

K ukládání dat jsou využity paměťi BRAM. Jejich počet je vždy násobkem dvanácti, což je nutné k správnému vyčítání dat. Princip toho vyčítáni je naznačen na obrázku 6.3. Po naadresování požadovananých dat je z paměti BRAM načten blok o rozměru 12×6 obrazových bodů. Tento blok je zarovnán tak, že adresa x je dělitelná šesti. Pro výběr požadovaných dat je tedy nutné použít ještě pětivstupé šestibitové multiplexory. Těch je zapotřebí celkem 36.

Kapacita paměti je dána velikosti ukládaného pruhu obrazu a také požadavkem na paralelní přístup do paměti. Počet použitých BRAM je vždy násobek dvanácti. V případě ukládání pruhu obrazu bez jeho zmenšených verzí stačí použít 12 pamětí BRAM. Do této paměti je možné uložit celkem 48 řádků obrazu, což je dvakrát více než je zapotřebí. Nadbytečná paměť však dobře slouží jako vyrovnávací buffer při práci s kamerou.

Pro detekci objektu různé velikosti je nutné velikost paměti výrazně zvětšit. Při použití navrhovaného postupu s jedním klasifikátorem byla vytvořena paměť využívající 48 BRAM s organizací 4608×32 obrazových bodů. Při použití druhého navrhovanáho postupu, s využitím více klasifikátorů pro detekci objektů různé velikosti, byla zapotřebí paměť využívající 24 BRAM s organizací 1536×48 bodů.



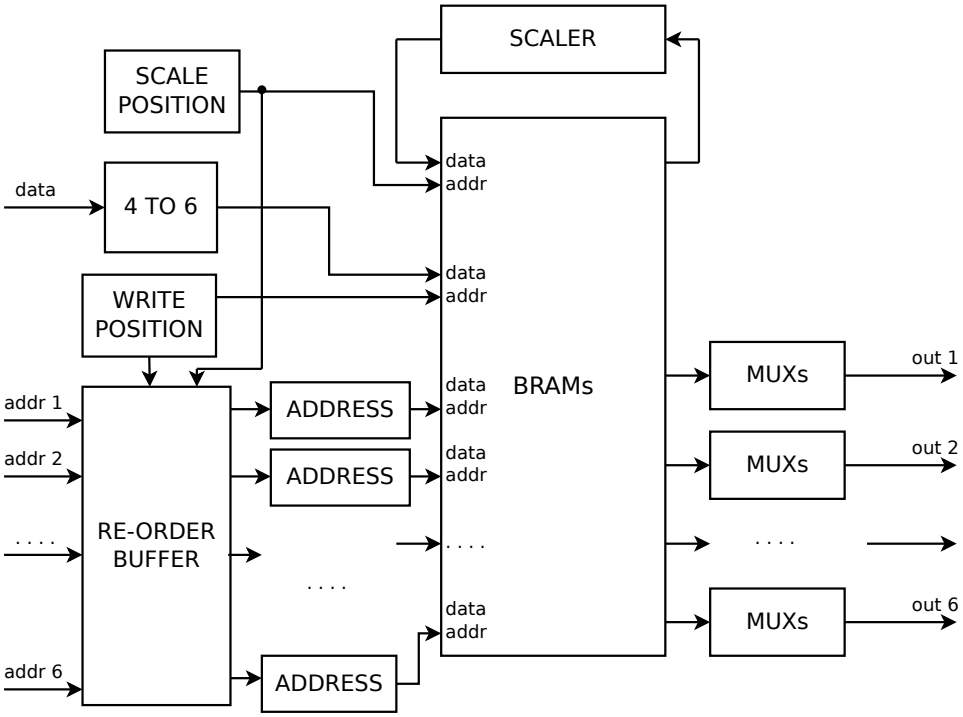
Obrázek 6.4: Zapojení paměti obrazu se současným vyčítáním dvou bloků a možností zmenšování obrazu.

Pro detekci objektů různé velikosti je nutné vytvářet v paměti zmenšené verze původního obrazu. Byla proto implementovaná jednotka pro změnu měřítka v poměru 1.2, což je standardní měřítko používaná pro detekci objektů různé velikosti. Tato jednotka využívá 487 Slice obvodů, což je 7% z celkového počtu. Nevýhodou této jednotky je nezarovnaný přístup do paměti při ukládání zmenšených výsledků. Je totiž nutné uložit blok dat o velikosti 5×5 obrazových bodů do paměti organizované po šesti hodnotách na řádku. Při zápisu je tedy nutné vyčíst minimálně dva bloky 6×6, hodnoty na určených pozicích přepsat a poté tyto bloky uložit zpátky. Pro zmenšení jednoho bloku o velikosti 6×6 na 5×5 je zapotřebí 7 hodinových taktů. Z toho důvodu byl použit druhý navrhovaný přístup. Ten provádí zmenšování vždy jen na polovinu. Implementovaná jednotka poté potřebuje jen 195

Slice obvodů, což je asi 3% celkového počtu. Její obrovskou výhodou je, že pro zmenšení jednoho bloku 12×6 na 6×3 potřebuje jen 3 hodinové takty. Je tedy více než 4.5 krát rychlejší než předchozí navrhovaná jednotka.

Pro zrychlení detekce objektů byly také implementovány dvě verze paměti umožňující paralelní běh detekce obrazu. První je naznačena na obrázku 6.4. Umožňuje vyčítání dvou nezávislých bloků z paměti. První port paměti je sdílen mezi více jednotkami a slouží pro načítání dat, vyčítání dat a pro připojení zmenšovací jednotky.

Další implementovaná verze je představena na obrázku 6.5. Využívá osmiportovou paměť, která je postavena z 48 BRAM. Přístup do této paměti neni nezávislý. To je dáno tím, že je vnitřně poskládána pomocí dvouportových BRAM. To dovoluje přistoupit k jedné adrese jen z dvou portů současně. Pro zamezení kolizí byla tedy vytvořena jednoduchá vyrovnávací paměť. Ta využívá osm front, z jejíž čela vybírá vhodné nekolidující adresy.



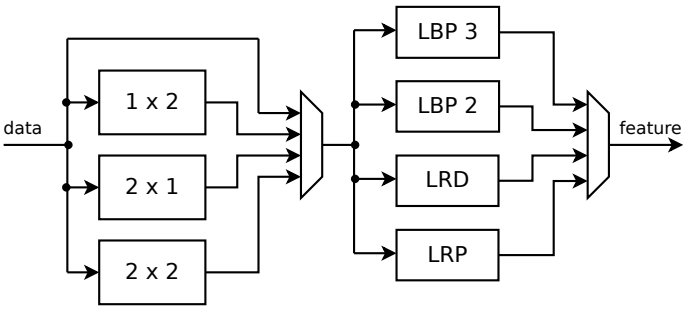
Obrázek 6.5: Zapojení paralelní architektury paměti obrazu pro vyčítání šesti bloku dat a možností zmenšování obrazu.

Architektura pro výpočet obrazových příznaků

Vyhodnocení obrazových příznaků je rozděleno do dvou modulů. První provádí výpočet odezvy s potřebným konvolučním jádrem a druhá slouží pro vyhodnocení samotného obrazového příznaku, jak je naznačeno na obrázku 6.6.

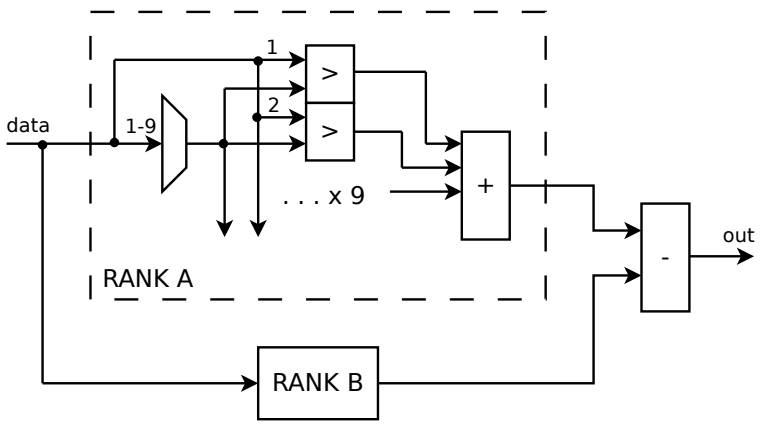
Výpočet odezvy je v tomto případě prakticky výpočet aritmetického průměru na příslušných hodnotách. Spočívá v sečtení těchto hodnot a následném vydělení jejich počtem. Toto dělení je nahrazeno rotací hodnoty. Pro výpočet konvoluce jsou využity sčítačky sestavené

z obvodů Slice. Provádí se vždy fitrování se všemi jádry a poté je vybrán potřebný výsledek.



Obrázek 6.6: Zapojení jednotky pro výpočet obrazového příznaku.

Detektor využívá obrazové příznaky typu LBP a LRF. Jsou implementovány LBP příznaky na okolí 2×2 a 3×3. Jejich výpočet je poměrně jednoduchý a spočívá v porovnání příslušných hodnot se středovou hodnotou. Dále jsou implementovány LRP a LRD příznaky na okolí 3×3. Pro výpočet těchto příznaku je nutné zjistit pořadí hodnot v daném okolí. Pořadí příslušné hodnoty je určeno z jejího porovnání s ostatními body okolí, jak je naznačeno na obrázku 6.7.



Obrázek 6.7: Zapojení jednotky pro výpočet LRD příznaku.

Náklady na zdroje pro výpočet jednotlivých obrazových příznaků jsou naznačeny v tabulce 6.2. Jednotka vždy provádí výpočet všech obrazových příznaků. Pro zpracování je vybrána jen potřebná hodnota. Některé části obvodů jsou mezi jednotkami vhodně sdíleny, což snižuje nároky na zdroje celého obvodu.

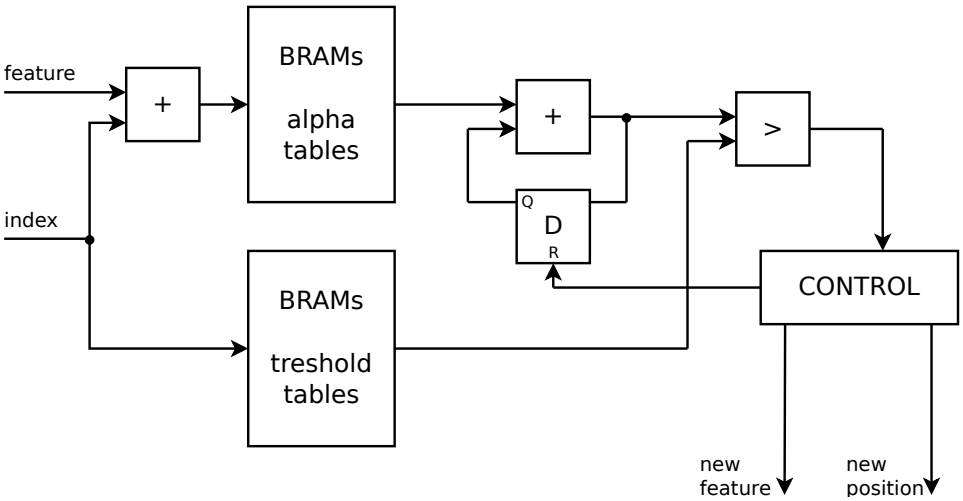
	Konvoluce	LBP 2×2	LBP 3×3	LRP 3×3	LRD 3×3	Celkový
Slice	97	7	13	46	41	172

Tabulka 6.2: Náklady na zdroje pro výpočet obrazových příznaků.

Vyhodnocení pozice

Schéma modulu pro vyhodnocení pozice je naznačeno na obrázku 6.8. Při vyhodnocení pozice je načtena váha klasifikátoru α. Ta je uložena v paměti BRAM. Adresa požadované váhy je dána součtem obrazového příznaku a indexu slabého klasifikátoru. Současně s váhou je načítán i příslušný práh. Váha klasifikátoru je přičtena k předchozímu součtu na vyhodnocované pozici a tento výsledek je porovnán s načteným prahem. Pokud je větší než práh, je generována instruce pro vyhodnocení dalšího klasifikáotru na příslušné pozici. Pokud je však menší, je pozici označena jako zamítnuta a dochází k vyhodnocení další pozice. Současně je vynulován i příslušný ukládaný součet. Počet těchto součtů je devět stejně jako hloubka zřetězené linky detektoru.

Kapacita paměti pro tabulky s váhou je omezena. Pokud neni použito vyhodnocení sousedních pozic je možné uložit až 512 tabulek, které používají šířku slova 9 bitů. Pro vyšší přesnost jsou implementovány i tabulky s šířkou slova 18 a 36 bitů, kterých je možno vytvořit 256 a 128. Pokud je pro zrychlení detekce použito vyhodnocení sousedních pozic, jsou použity tabulky s šířkou slova 72 bitů, kterých je možno implementovet maximáně 128.



Obrázek 6.8: Zapojení jednotky pro vyhodnocení pozice klasifikátorem.

Rozhraní detektoru

Detektor komunikuje pomocí rozhraní vytvořené Martinem Musilem v [16]. Toto rozhrani obsahuje moduly pro práci s kamerou, pamětí RAM a sběrnicí PCIe pro komunikaci s počítačem.

Na vstup detektoru je přiveden proud dat z kamery. Je použito synchronní rozhraní se signálem pro potvrzení dat. Ke kameře je naopak přiveden signál, který informuje o připravenosti detektoru.

Výsledky detekce jsou přivedeny na počítač. K tomu je využito rozhraní s PCIe. Jsou přenášeny pouze informace o pozitivně detekované pozici. Na počítač je zaslána poloha v obraze, příslušné měřítko a hodnota součtu detekovaná pozice.

Hodnoty tabulek s váhou α, prahů a instrukcí jsou uloženy do pamětí BRAM při překladu.

Je však také implementován modul, který by bylo možné použít pro jejich načítání z paměti RAM.

6.3 Testování detektoru

Správná funkčnost detektoru byla testována pomocí simulačního nastroje iSim od Xilinxu. Byla testována funkčnost a vlastnosti jednotlivých modulů a detektoru jako celku.

Ke každému implementovanému modulu je vytvořen příslušný testovací soubor. Při testování jsou generovány vhodné vstupní vektory, které jsou vyhodnoceny testovaným modulem a výsledek je porovnán s očekavaným výsledkem. Takto byl zvláš otestován každý modul.

Byl také otestován správný chod celého detektoru. Pomocí testovacího souboru byl načten kousek obrázku z testovací sady CMU, který obsahoval lidské tváře. Byly načteny příslušné tabulky s váhou, prahy a byly vytvořeny instrukce pro řízení klasifikátoru. Následně byla otestována správná funkčnost. Detektor vyhodnotil vstupní obrázek správně dle očekávání a na příslušných pozicích úspěšně detekoval lidské tváře v obrázku. Z důvodu časové náročnosti bylo provedeno těchto testů pouze pět. Průměrné naměřené hodnoty jsou uvedeny v tabulce 6.3.

	Příznaků za takt	Snímků bez zmenšení	Snímků s zmenšením
verze 1	0.98	100	N
verze 2	1.76	180	N
verze 3	1.76	180	57
verze 4	3.8	390	124

Tabulka 6.3: Naměřená výkonost jednotlivých verzí detektoru.

Z naměřených hodnot vyplývá, že každá verze detektoru umožňuje detekci objektů jedné velikosti ve videu z připojené kamery. Současně je možné vidět nárůst výkonosti u paralelních verzí detektoru. Tyto paralelní verze mají dostatečný výkon pro detekci objektů různé velikosti. Poslední implementovaná verze detektoru má dokonce takový výkon, že umožňuje detekci dvou tříd objektů současně.

Pro ověření možnosti realizace byl vytvořen detektor objektů pracující nad tokem dat z kamery na vývojovém kitu. Tento detektor však nebyl odladěn a testován z důvodu dlouhodobé nedostupnosti kamery. Ta je použita na jiném projektu. Po dodání kamery bude správný chod detektoru ověřen.

Byl tedy vytvořen jiný jednoduchý experiment, který ověřuje chování subsystému detektoru. Spočíval ve vyhodnocování obrazových příznaků nad tokem dat uloženým v paměti BRAM. Funkčnost tohoto experimentu byla úspěšně odzkoušena na vývojovém kitu.

Kapitola 7

Závěr

Cílem diplomové práce bylo navrhnout, implementovat a otestovat detektor objektů ve videu. Tyto cíle se podařilo splnit. Byly také kompletně splněny všechny body stanovené v zadání práce. Byla prostudována literatura týkající se zpracování obrazu a tvorby systémů na FPGA. Nejdůležitější nastudované informace jsou představeny v kapitolách 2 a 3 této práce. Pro realizaci byl následně vybrán algoritmus detekce objektů ve videu, který je vhodný případ výpočetně náročných algoritmů zpracování obrazu. V kapitole 5 byl navržen vhodný způsob implementace a byly představeny úpravy stávajícího algoritmu detekce objektů pro použití na FPGA. Samotná realizace detektoru je popsána v kapitole 6. Součástí teto kapitoly je také testování jednotlivých částí a celku výsledného detektoru.

Navržený systém implementovaný na FPGA umožňuje detekci objektů ve videu v reálném čase. Má dostatečný výkon pro souběžnou detekci až dvou různých tříd objektů. Umožňuje detekci objektů různé velikosti, k čemuž využívá nově navržený přístup, který je paměťově čtyřikrát méně náročnější než současné jednočipové řešení. Ke své činnosti nepotřebuje žádné externí výpočetní jednotky. Systém jako první využívá algoritmy pro zrychlení detekce, které se doposud používaly pouze u procesorových systémů. Je také navržen a implementován nový efektivní způsob paralelizace detektoru, který přináší výrazné zrychlení jeho práce. Navržený detektor byl úspěšně implementován a odsimulován. Některé jeho části byly také vyzkoušeny na vývojovém kitu.

V dalším vývoji plánuji otestovat kompletní systém na vývojovém kitu. Bylo by dobré ověřit chování detektoru při použití předzpracovací jednotky, jejíž parametry nebyly v době testování známy. Pro možnost uplatnění systému v praxi by bylo vhodné zapracovat do návrhu některé principy tvorby úsporných zařízení. Zajímavé by také určitě bylo použít navržený detektor v některé reálné aplikaci. Jednou z těchto aplikací by mohlo být například využití detektoru v robotickém systému.

Literatura

[1] Ahilan, A.; James, E.: Design and implementation of real time car theft detection in FPGA. In *Third International Conference on Advanced Computing (ICoAC)*, IEEE Computer Society, 2011, ISBN 978-1-4673-0670-6, s. 353 – 358.

[2] Augusteijn, M. F.; Skufca, T. L.: *Identification of Human Faces through Texture-Based Feature Recognition and Neural Network Technology.* IEEE Neural Networks, 1993, 0-7803-0999-5.

[3] Cho, J.; Mirzaei, S.; Oberg, J.; aj.: Fpga-based face detection system using Haar classifiers. In *International Symposium on Field Programmable Gate Arrays*, ACM New York, 2009, ISBN 978-1-60558-410-2, s. 103 – 112.

[4] Cortes, C.; Vapnik, V.: *Support-vector networks.* Machine Learning, Volume 20, Number 3, 2005, 0-7803-0999-5.

[5] Freund, Y.; Schapire, R. E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In *Journal of Computer and System Sciences*, Springer-Verlag, 1997, ISBN 978-953-7619-90-9.

[6] Haar, A.: *Zur Theorie der orthogonalen Funktionensysteme.* Mathematische Annalen, 1910.

[7] Herout, A.; Hradiš, M.; Zemčík, P.: EnMS: Early non-Maxima Suppression. *Pattern Analysis and Applications*, ročník 2011, č. 1111, 2011: str. 10, ISSN 1433-7541.

[8] Herout, A.; Zemčík, P.; Hradiš, M.; aj.: *Low-Level Image Features for Real-Time Object Detection*. IN-TECH Education and Publishing, 2010, ISBN 978-953-7619-90-9, s. 111–136.

[9] Hradiš, M.: *AdaBoost in Computer Vision.* Diplomová práce, VUT FIT, Brno, 2007.

[10] Hradiš, M.; Herout, A.; Zemčík, P.: Local Rank Patterns - Novel Features for Rapid Object Detection. In *Proceedings of International Conference on Computer Vision and Graphics 2008*, číslo 12 in Lecture Notes in Computer Science, Springer Verlag, 2008, ISSN 0302-9743, s. 1–12.

[11] Huang, C.; Vahid, F.: Scalable object detection accelerators on FPGAs using custom design space exploration. In *IEEE Symposium on Application Specific Processors (SASP)*, IEEE Computer Society, 2011, ISBN 978-1-4577-1212-8, s. 115–121.

[12] Juránek, R.; Hradiš, M.; Zemčík, P.: *Real-time Algorithms of Object Detection using Classifiers*. InTech - Open Access Publisher, 2012, ISBN 978-953-510-510-7, s. 1–22.

[13] Kyrkou, C.; Theocharides, T.: "A Flexible Parallel Hardware Architecture for AdaBoost-Based Real-Time Object Detection". In *Very Large Scale Integration (VLSI) Systems*, IEEE Computer Society, 2011, ISSN 1063-8210, s. 1034 – 1047.

[14] Lai, H.-C.; Savvides, M.; Chen, T.: Proposed FPGA Hardware Architecture for High Frame Rate (?100 fps) Face Detection Using Feature Cascade Classifiers. In *Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007*, IEEE Computer Society, 2007, ISBN 978-1-4244-1596-0, s. 1 – 6.

[15] Michie, D.; Spiegelhalter, D. J.; Taylor, C.: Machine Learning, Neural and Statistical Classification. 1994.

[16] Musil, M.; Musil, P.: PCI Express Interface for Signal and Video Processing. In *Student EEICT*, 2012, str. 3.

[17] Mäenpää, T.; Pietikäinen, M.: *Texture Analysis with Local Binary Patterns*. University of Oulu, Finland, 2004.

[18] Neoh, H.; Hazanchuk, A.: Adaptive Edge Detection for Real-Time Video Processing using FPGAs. *Global Signal Processing*, 2004.

[19] Pinkler, J.; Poupa, M.: *Číslicové systémy a jazyk VHDL*. BEN - Technická literatura, 2006, iSBN 80-7300-198-5.

[20] Polok, L.; Herout, A.; Zemčík, P.; aj.: "Local Rank Differences" Image Feature Implemented on GPU. In *Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems*, Lecture Notes In Computer Science; Vol. 5259, Springer Verlag, 2008, ISBN 978-3-540-88457-6, s. 170–181.

[21] Šochman, J.; Matas, J.: WaldBoost - Learning for Time Constrained Sequential Detection. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, USA: IEEE Computer Society, June 2005, ISBN 0-7695-2372-2, s. 150–157.

[22] Theocharides, T.; Link, G.; Vijaykrishnan, N.; aj.: Embedded hardware face detection. In *17th International Conference on VLSI Design*, IEEE Computer Society, 2004, ISBN 0-7695-2072-3, s. 133 – 138.

[23] Viola, P.; Jones, M.: *Robust Real-time Object Detection*. SCTV Vancouver, 2001.

[24] Wang, X.; Gong, H.; Zhang, H.; aj.: *Palmprint Identification using Boosting Local Binary Pattern*. International Conference on Pattern Recognition, Hong Kong, 2006.

[25] WWW stránky: Dokumentace k SP 605. `http://www.xilinx.com/products/boards-and-kits/EK-S6-SP605-G.htm`, 2012-01-10 [cit. 2010-01-10].

[26] WWW stránky: Dokumentace k Spartan-6 FPGA Family. `http://www.xilinx.com/products/silicon-devices/fpga/spartan-6/index.htm`, 2012-01-10 [cit. 2010-01-10].

[27] WWW stránky: Dokumentace k Virtex-7 FPGA Family. `http://www.xilinx.com/products/silicon-devices/fpga/virtex-7/index.htm`, 2012-01-10 [cit. 2010-01-10].

[28] Yang, G.; Huang, T. S.: *Human face detection in a complex background*. Pattern Recognition, 1994.

[29] Yuille, A. L.; Hallinan, P. W.; Cohen, D. S.: *Feature Extraction from Faces Using Deformable Templates*. IJCV, 1992.

[30] Yutong, Z.; Guosheng, Y.; Licheng, W.: Fast Face Detection in Field Programmable Gate Array. In *Digital Manufacturing and Automation (ICDMA)*, IEEE Computer Society, 2010, ISBN 978-0-7695-4286-7, s. 719 – 723.

[31] Zemčík, P.: Hardware Acceleration of Graphics and Imaging Algorithms using FPGAs. In *Proceedings of SCCG*, Slovak University of Technology in Bratislava, 2002, ISBN 1-58113-608-0, str. 8.

[32] Zemčík, P.; Hradiš, M.; Herout, A.: Local Rank Differences - Novel Features for Image. In *Poster SCCG 2007*, 2007, str. 12.

[33] Zemčík, P.; Hradiš, M.; Herout, A.: Exploiting neighbors for faster scanning window detection in images. In *ACIVS 2010*, LNCS 6475, Springer Verlag, 2010, ISBN 978-3-642-17690-6, str. 12.

[34] Zemčík, P.; Žádník, M.: Adaboost Engine. In *Proceedings of FPL 2007*, IEEE Computer Society, 2007, ISBN 1-4244-1060-6, str. 5.